Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32H7A3/7B3 and STM32H7B0 microcontroller memory and peripherals.

The STM32H7A3/7B3 and STM32H7B0 are microcontrollers with different memory sizes, packages and peripherals.

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

For information on the Arm® Cortex®-M7 with FPU cores, please refer to the corresponding Arm Technical Reference Manuals.

Related documents

- Cortex®-M7 programming manual (PM0253).
- STM32H7A3xl/G, STM32H7B3xl and STM32H7B0xB datasheets.
## Contents

1. **Documentation conventions** ......................................................... 97  
   1.1 General information ................................................................. 97  
   1.2 List of abbreviations for registers .............................................. 97  
   1.3 Glossary ...................................................................................... 98  
   1.4 Availability of peripherals ......................................................... 98  
   1.5 Availability of security features .................................................. 98  

2. **Memory and bus architecture** ..................................................... 99  
   2.1 System architecture ...................................................................... 99  
      2.1.1 Bus matrices ......................................................................... 102  
      2.1.2 TCM buses ........................................................................... 102  
      2.1.3 Bus-to-bus bridges ................................................................. 102  
      2.1.4 Domain and inter-domain buses ............................................. 103  
      2.1.5 CPU buses ............................................................................. 103  
      2.1.6 Bus master peripherals .......................................................... 104  
      2.1.7 Clocks to functional blocks .................................................... 105  
   2.2 AXI interconnect matrix (AXIM) ................................................ 105  
      2.2.1 AXI introduction ..................................................................... 105  
      2.2.2 AXI interconnect main features .............................................. 105  
      2.2.3 AXI interconnect functional description .................................. 106  
      2.2.4 AXI interconnect registers ...................................................... 108  
      2.2.5 AXI interconnect register map ................................................ 117  
   2.3 Memory organization .................................................................... 128  
      2.3.1 Introduction ............................................................................ 128  
      2.3.2 Register boundary addresses ................................................ 129  
   2.4 Embedded SRAM .......................................................................... 134  
   2.5 Flash memory overview ............................................................... 135  
   2.6 Boot configuration ........................................................................ 135  

3. **RAM ECC monitoring (RAMECC)** ............................................ 137  
   3.1 Introduction .................................................................................. 137  
   3.2 RAMECC main features ............................................................... 137  
   3.3 RAMECC functional description .................................................. 137
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.1</td>
<td>RAMECC block diagram</td>
<td>137</td>
</tr>
<tr>
<td>3.3.2</td>
<td>RAMECC internal signals</td>
<td>139</td>
</tr>
<tr>
<td>3.3.3</td>
<td>RAMECC monitor mapping</td>
<td>139</td>
</tr>
<tr>
<td>3.4</td>
<td>RAMECC registers</td>
<td>140</td>
</tr>
<tr>
<td>3.4.1</td>
<td>RAMECC interrupt enable register (RAMECC_IER)</td>
<td>140</td>
</tr>
<tr>
<td>3.4.2</td>
<td>RAMECC monitor x configuration register (RAMECC_MxCR)</td>
<td>141</td>
</tr>
<tr>
<td>3.4.3</td>
<td>RAMECC monitor x status register (RAMECC_MxSR)</td>
<td>141</td>
</tr>
<tr>
<td>3.4.4</td>
<td>RAMECC monitor x failing address register (RAMECC_MxFAR)</td>
<td>142</td>
</tr>
<tr>
<td>3.4.5</td>
<td>RAMECC monitor x failing data low register (RAMECC_MxFDRL)</td>
<td>142</td>
</tr>
<tr>
<td>3.4.6</td>
<td>RAMECC monitor x failing data high register (RAMECC_MxFDRH)</td>
<td>143</td>
</tr>
<tr>
<td>3.4.7</td>
<td>RAMECC monitor x failing ECC error code register</td>
<td>143</td>
</tr>
<tr>
<td></td>
<td>RAMECC_MxFECR</td>
<td></td>
</tr>
<tr>
<td>3.4.8</td>
<td>RAMECC register map</td>
<td>144</td>
</tr>
<tr>
<td>4</td>
<td>Embedded Flash memory (FLASH)</td>
<td>145</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>145</td>
</tr>
<tr>
<td>4.2</td>
<td>FLASH main features</td>
<td>145</td>
</tr>
<tr>
<td>4.3</td>
<td>FLASH functional description</td>
<td>146</td>
</tr>
<tr>
<td>4.3.1</td>
<td>FLASH block diagram</td>
<td>146</td>
</tr>
<tr>
<td>4.3.2</td>
<td>FLASH internal signals</td>
<td>147</td>
</tr>
<tr>
<td>4.3.3</td>
<td>FLASH architecture and integration in the system</td>
<td>147</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Flash memory architecture and usage</td>
<td>149</td>
</tr>
<tr>
<td>4.3.5</td>
<td>FLASH system performance enhancements</td>
<td>154</td>
</tr>
<tr>
<td>4.3.6</td>
<td>FLASH data protection schemes</td>
<td>154</td>
</tr>
<tr>
<td>4.3.7</td>
<td>Overview of FLASH operations</td>
<td>155</td>
</tr>
<tr>
<td>4.3.8</td>
<td>FLASH read operations</td>
<td>156</td>
</tr>
<tr>
<td>4.3.9</td>
<td>FLASH program operations</td>
<td>160</td>
</tr>
<tr>
<td>4.3.10</td>
<td>FLASH erase operations</td>
<td>164</td>
</tr>
<tr>
<td>4.3.11</td>
<td>FLASH parallel operations</td>
<td>167</td>
</tr>
<tr>
<td>4.3.12</td>
<td>Flash memory error protections</td>
<td>167</td>
</tr>
<tr>
<td>4.3.13</td>
<td>FLASH one-time programmable area</td>
<td>169</td>
</tr>
<tr>
<td>4.3.14</td>
<td>FLASH read-only area</td>
<td>170</td>
</tr>
<tr>
<td>4.3.15</td>
<td>Flash bank and register swapping (STM32H7A3/7B3 only)</td>
<td>171</td>
</tr>
<tr>
<td>4.3.16</td>
<td>FLASH reset and clocks</td>
<td>175</td>
</tr>
<tr>
<td>4.4</td>
<td>FLASH option bytes</td>
<td>175</td>
</tr>
<tr>
<td>4.4.1</td>
<td>About option bytes</td>
<td>175</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Option byte loading</td>
<td>176</td>
</tr>
</tbody>
</table>
4.4.3 Option byte modification ........................................... 176
4.4.4 Option bytes overview ............................................ 178
4.4.5 Description of user and system option bytes ................. 182
4.4.6 Description of data protection option bytes ................. 183
4.4.7 Description of boot address option bytes ..................... 184

4.5 FLASH protection mechanisms ....................................... 184
4.5.1 FLASH configuration protection ................................. 184
4.5.2 Write protection .................................................. 186
4.5.3 Readout protection (RDP) ....................................... 187
4.5.4 Proprietary code readout protection (PCROP) ............... 191
4.5.5 Secure access mode (STM32H7B0 and STM32H7B3 only) .... 192

4.6 FLASH low-power modes ............................................. 194
4.6.1 Introduction ....................................................... 194
4.6.2 Managing the FLASH domain switching to DStop or Standby 194

4.7 FLASH error management ............................................ 195
4.7.1 Introduction ....................................................... 195
4.7.2 Write protection error (WRPERR) ............................. 195
4.7.3 Programming sequence error (PGSERR) ......................... 196
4.7.4 Strobe error (STRBERR) ....................................... 197
4.7.5 Inconsistency error (INCERR) ................................ 197
4.7.6 Error correction code error (SNECCERR/DBECCERR) ....... 198
4.7.7 Read protection error (RDPERR) .............................. 198
4.7.8 Read secure error (RDSERR) .................................. 199
4.7.9 CRC read error (CRCRDERR) ................................ 199
4.7.10 Option byte change error (OPTCHANGEERR) .............. 200
4.7.11 Miscellaneous HardFault errors ............................... 200

4.8 FLASH interrupts .................................................... 200

4.9 FLASH registers .................................................... 203
4.9.1 FLASH access control register (FLASH_ACR) .............. 203
4.9.2 FLASH key register for bank 1 (FLASH_KEYR1) ............. 203
4.9.3 FLASH option key register (FLASH_OPTKEYR) ............. 204
4.9.4 FLASH control register for bank 1 (FLASH_CR1) .......... 204
4.9.5 FLASH status register for bank 1 (FLASH_SR1) .............. 208
4.9.6 FLASH clear control register for bank 1 (FLASH_CCR1) ... 211
4.9.7 FLASH option control register (FLASH_OPTCR) ............ 212
4.9.8 FLASH option status register (FLASH_OPTSR_CUR) ....... 213
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.9.9</td>
<td>FLASH option status register (FLASH_OPTSR_PRG)</td>
</tr>
<tr>
<td>4.9.10</td>
<td>FLASH option clear control register (FLASH_OPTCCR)</td>
</tr>
<tr>
<td>4.9.11</td>
<td>FLASH protection address for bank 1 (FLASH_PRAR_CUR1)</td>
</tr>
<tr>
<td>4.9.12</td>
<td>FLASH protection address for bank 1 (FLASH_PRAR_PRG1)</td>
</tr>
<tr>
<td>4.9.13</td>
<td>FLASH secure address for bank 1 (FLASH_SCAR_CUR1)</td>
</tr>
<tr>
<td>4.9.14</td>
<td>FLASH secure address for bank 1 (FLASH_SCAR_PRG1)</td>
</tr>
<tr>
<td>4.9.15</td>
<td>FLASH write sector group protection for bank 1 (FLASH_WPSGN_CUR1R)</td>
</tr>
<tr>
<td>4.9.16</td>
<td>FLASH write sector group protection for bank 1 (FLASH_WPSGN_PRG1R)</td>
</tr>
<tr>
<td>4.9.17</td>
<td>FLASH register boot address (FLASH_BOOT_CURR)</td>
</tr>
<tr>
<td>4.9.18</td>
<td>FLASH register boot address (FLASH_BOOT_PRGR)</td>
</tr>
<tr>
<td>4.9.19</td>
<td>FLASH CRC control register for bank 1 (FLASH_CRCRCR1)</td>
</tr>
<tr>
<td>4.9.20</td>
<td>FLASH CRC start address register for bank 1 (FLASH_CRCSSAADD1R)</td>
</tr>
<tr>
<td>4.9.21</td>
<td>FLASH CRC end address register for bank 1 (FLASH_CRCSEADD1R)</td>
</tr>
<tr>
<td>4.9.22</td>
<td>FLASH CRC data register (FLASH_CRCDATAR)</td>
</tr>
<tr>
<td>4.9.23</td>
<td>FLASH ECC fail address for bank 1 (FLASH_ECC_FA1R)</td>
</tr>
<tr>
<td>4.9.24</td>
<td>FLASH OTP block lock (FLASH_OTPBL_CUR)</td>
</tr>
<tr>
<td>4.9.25</td>
<td>FLASH OTP block lock (FLASH_OTPBL_PRG)</td>
</tr>
<tr>
<td>4.9.26</td>
<td>FLASH key register for bank 2 (FLASH_KEYR2)</td>
</tr>
<tr>
<td>4.9.27</td>
<td>FLASH control register for bank 2 (FLASH_CR2)</td>
</tr>
<tr>
<td>4.9.28</td>
<td>FLASH status register for bank 2 (FLASH_SR2)</td>
</tr>
<tr>
<td>4.9.29</td>
<td>FLASH clear control register for bank 2 (FLASH_CCR2)</td>
</tr>
<tr>
<td>4.9.30</td>
<td>FLASH protection address for bank 2 (FLASH_PRAR_CUR2)</td>
</tr>
<tr>
<td>4.9.31</td>
<td>FLASH protection address for bank 2 (FLASH_PRAR_PRG2)</td>
</tr>
<tr>
<td>4.9.32</td>
<td>FLASH secure address for bank 2 (FLASH_SCAR_CUR2)</td>
</tr>
<tr>
<td>4.9.33</td>
<td>FLASH secure address for bank 2 (FLASH_SCAR_PRG2)</td>
</tr>
<tr>
<td>4.9.34</td>
<td>FLASH write sector group protection for bank 2 (FLASH_WPSGN_CUR2R)</td>
</tr>
<tr>
<td>4.9.35</td>
<td>FLASH write sector group protection for bank 2 (FLASH_WPSGN_PRG2R)</td>
</tr>
<tr>
<td>4.9.36</td>
<td>FLASH CRC control register for bank 2 (FLASH_CRCRCR2)</td>
</tr>
<tr>
<td>4.9.37</td>
<td>FLASH CRC start address register for bank 2 (FLASH_CRCSSAADD2R)</td>
</tr>
<tr>
<td>4.9.38</td>
<td>FLASH CRC end address register for bank 2 (FLASH_CRCSEADD2R)</td>
</tr>
<tr>
<td>4.9.39</td>
<td>FLASH ECC fail address for bank 2 (FLASH_ECC_FA2R)</td>
</tr>
</tbody>
</table>
5 Secure memory management (SMM) ........................................ 250
  5.1 Introduction ........................................................... 250
  5.2 Glossary .............................................................. 250
  5.3 Secure access mode .................................................. 251
    5.3.1 Associated features ........................................... 252
    5.3.2 Boot state machine ........................................... 252
    5.3.3 Secure access mode configuration ........................... 253
  5.4 Root secure services (RSS) ....................................... 254
    5.4.1 Secure area setting service ................................ 254
    5.4.2 Secure area exiting service ................................ 254
    5.4.3 OTFDEC encryption service ................................. 255
  5.5 Secure user software .............................................. 255
    5.5.1 Access rules .................................................... 255
    5.5.2 Setting secure user memory areas ........................... 255
  5.6 Summary of Flash protection mechanisms ....................... 256

6 Power control (PWR) .................................................. 257
  6.1 Introduction .......................................................... 257
  6.2 PWR main features .................................................. 257
  6.3 PWR block diagram .................................................. 258
  6.4 Power supplies ...................................................... 260
    6.4.1 System supply startup ...................................... 265
    6.4.2 Core domain ................................................... 268
    6.4.3 PWR external supply ........................................ 270
    6.4.4 Backup domain ................................................ 270
    6.4.5 VBAT battery charging ..................................... 272
    6.4.6 Analog supply ................................................ 272
    6.4.7 USB regulator ................................................ 273
  6.5 Power supply supervision ......................................... 274
    6.5.1 Power-on reset (POR)/power-down reset (PDR) ............ 274
    6.5.2 Brownout reset (BOR) ...................................... 275
    6.5.3 Programmable voltage detector (PVD) ...................... 275
    6.5.4 Analog voltage detector (AVD) ............................. 277
    6.5.5 Battery voltage thresholds ................................ 278
6.5.6 Temperature thresholds ............................................ 279
6.5.7 $V_{\text{CORE}}$ maximum voltage level detector ................. 279

6.6 Power management .................................................. 280
6.6.1 Operating modes .................................................. 281
6.6.2 Voltage scaling ................................................... 284
6.6.3 Power control modes ............................................. 285
6.6.4 Power management examples .................................. 287

6.7 Low-power modes .................................................. 295
6.7.1 Slowing down system clocks ................................... 295
6.7.2 Controlling peripheral clocks ................................ 295
6.7.3 Entering low-power modes ................................... 295
6.7.4 Exiting from low-power modes ................................. 296
6.7.5 CSleep mode ...................................................... 296
6.7.6 CStop mode ....................................................... 297
6.7.7 DStop and DStop2 modes ....................................... 298
6.7.8 Stop mode ......................................................... 300
6.7.9 Standby mode ..................................................... 303
6.7.10 Monitoring low-power modes ................................. 304

6.8 PWR registers ....................................................... 306
6.8.1 PWR control register 1 (PWR_CR1) ............................. 306
6.8.2 PWR control status register 1 (PWR_CSR1) .................... 308
6.8.3 PWR control register 2 (PWR_CR2) ............................. 309
6.8.4 PWR control register 3 (PWR_CR3) ............................. 310
6.8.5 PWR CPU control register (PWR_CPCR) ....................... 312
6.8.6 PWR SmartRun domain control register (PWR_SRDCR) .... 313
6.8.7 PWR wakeup clear register (PWR_WKUPCR) .................. 314
6.8.8 PWR wakeup flag register (PWR_WKUPFR) .................... 315
6.8.9 PWR wakeup enable and polarity register (PWR_WKUPEPR) . 315
6.8.10 PWR register map .............................................. 316

7 Low-power SRD domain .............................................. 318
7.1 Introduction ......................................................... 318
7.2 EXTI, RCC and PWR interconnections ........................... 318
7.2.1 Interrupts and wakeup ......................................... 320
7.2.2 Block interactions ............................................... 320
7.2.3 Role of SRD domain DMAMUX2 ............................. 321
7.3 Low-power application example based on LPUART1 transmission ........................................ 322
7.3.1 Memory retention ................................................................. 322
7.3.2 Memory-to-peripheral transfer using LPUART1 interface ............................... 322
7.3.3 Overall description of the low-power application example based on LPUART1 transmission ........................................ 327
7.3.4 Alternate implementations ......................................................... 328
7.4 Other low-power applications ...................................................... 329

8 Reset and clock control (RCC) ....................................................... 330
8.1 RCC main features ................................................................. 330
8.2 RCC block diagram ................................................................. 331
8.3 RCC pins and internal signals ...................................................... 331
8.4 RCC reset block functional description ........................................ 333
  8.4.1 Power-on/off reset ............................................................ 333
  8.4.2 System reset ................................................................. 334
  8.4.3 Local resets ................................................................. 335
  8.4.4 Reset source identification .................................................. 335
  8.4.5 Low-power mode security reset (lpwr_rst) .................................. 336
  8.4.6 Backup domain reset ......................................................... 337
  8.4.7 Power-on and wakeup sequences ........................................... 337
8.5 RCC clock block functional description ........................................ 340
  8.5.1 Clock naming convention .................................................... 342
  8.5.2 Oscillators description ....................................................... 342
  8.5.3 Clock security system (CSS) ................................................ 348
  8.5.4 Clock output generation (MCO1/MCO2) ...................................... 349
  8.5.5 PLL description ............................................................ 349
  8.5.6 System clock (sys_ck) ....................................................... 354
  8.5.7 Handling clock generators in Stop and Standby modes ......................... 356
  8.5.8 Kernel clock selection ....................................................... 357
  8.5.9 General clock concept overview ............................................. 369
  8.5.10 Peripheral allocation ....................................................... 373
  8.5.11 Peripheral clock gating control ............................................ 374
  8.5.12 CPU and bus matrix clock gating control .................................... 379
8.6 RCC interrupts ................................................................. 380
8.7 RCC registers ................................................................. 381
  8.7.1 RCC source control register (RCC_CR) ...................................... 381
### Contents

8.7.2 RCC HSI calibration register (RCC_HSICFGR) .......................... 385
8.7.3 RCC clock recovery RC register (RCC_CRRCR) ........................ 385
8.7.4 RCC CSI calibration register (RCC_CSI CFGR) ......................... 386
8.7.5 RCC clock configuration register (RCC_CFGR) ....................... 387
8.7.6 RCC CPU domain clock configuration register 1 (RCC_CDCFGR1) 389
8.7.7 RCC CPU domain clock configuration register 2 (RCC_CDCFGR2) 391
8.7.8 RCC SmartRun domain clock configuration register (RCC_SRDCCFGR) 391
8.7.9 RCC PLLs clock source selection register (RCC_PLLCKSELR) .... 392
8.7.10 RCC PLLs configuration register (RCC_PLLCFGR) .................. 394
8.7.11 RCC PLL1 dividers configuration register (RCC_PLL1DIVR) .... 396
8.7.12 RCC PLL1 fractional divider register (RCC_PLL1FRACR) ........ 398
8.7.13 RCC PLL2 dividers configuration register (RCC_PLL2DIVR) .... 398
8.7.14 RCC PLL2 fractional divider register (RCC_PLL2FRACR) ....... 400
8.7.15 RCC PLL3 dividers configuration register (RCC_PLL3DIVR) .... 400
8.7.16 RCC PLL3 fractional divider register (RCC_PLL3FRACR) ....... 402
8.7.17 RCC CPU domain kernel clock configuration register (RCC_CDCCIPR) 402
8.7.18 RCC CPU domain kernel clock configuration register (RCC_CDCCIP1R) 403
8.7.19 RCC CPU domain kernel clock configuration register (RCC_CDCCIP2R) 406
8.7.20 RCC SmartRun domain kernel clock configuration register (RCC_SRDCCIPR) 408
8.7.21 RCC clock source interrupt enable register (RCC_CIER) ......... 410
8.7.22 RCC clock source interrupt flag register (RCC_CIFR) .......... 411
8.7.23 RCC clock source interrupt clear register (RCC_CICR) ........ 413
8.7.24 RCC Backup domain control register (RCC_BDCR) ............. 415
8.7.25 RCC clock control and status register (RCC_CSR) ............. 417
8.7.26 RCC AHB3 reset register (RCC_AHB3RSTR) ...................... 417
8.7.27 RCC AHB1 peripheral reset register (RCC_AHB1RSTR) ......... 419
8.7.28 RCC AHB2 peripheral reset register (RCC_AHB2RSTR) ......... 420
8.7.29 RCC AHB4 peripheral reset register (RCC_AHB4RSTR) ......... 421
8.7.30 RCC APB3 peripheral reset register (RCC_APB3RSTR) ......... 423
8.7.31 RCC APB1 peripheral reset register (RCC_APB1RSTR) ......... 423
8.7.32 RCC APB1 peripheral reset register (RCC_APB1HRSTR) ...... 426
8.7.33 RCC APB2 peripheral reset register (RCC_APB2RSTR) ......... 427
8.7.34 RCC APB4 peripheral reset register (RCC_APB4RSTR) ......... 429
8.7.35 RCC SmartRun domain autonomous mode register (RCC_SRDAMR) ................................................. 431
8.7.36 RCC AXI clocks gating enable register (RCC_CKGAENR) ......................................................... 434
8.7.37 RCC reset status register (RCC_RSR) ....................................................................................... 437
8.7.38 RCC AHB3 clock register (RCC_AHB3ENR) ............................................................................. 439
8.7.39 RCC AHB1 clock register (RCC_AHB1ENR) ............................................................................. 440
8.7.40 RCC AHB2 clock register (RCC_AHB2ENR) ............................................................................. 442
8.7.41 RCC AHB4 Clock Register (RCC_AHB4ENR) ........................................................................ 443
8.7.42 RCC APB3 clock register (RCC_APB3ENR) ............................................................................. 445
8.7.43 RCC APB1 clock register (RCC_APB1LENR) ........................................................................... 446
8.7.44 RCC APB1 clock register (RCC_APB1HENR) ........................................................................... 449
8.7.45 RCC APB2 clock register (RCC_APB2ENR) ............................................................................. 450
8.7.46 RCC APB4 clock register (RCC_APB4ENR) ............................................................................. 452
8.7.47 RCC AHB3 sleep clock register (RCC_AHB3LPENR) ................................................................. 454
8.7.48 RCC AHB1 sleep clock register (RCC_AHB1LPENR) ................................................................. 457
8.7.49 RCC AHB2 sleep clock register (RCC_AHB2LPENR) ................................................................. 458
8.7.50 RCC AHB4 sleep clock register (RCC_AHB4LPENR) ................................................................. 459
8.7.51 RCC APB3 sleep clock register (RCC_APB3LPENR) ................................................................. 461
8.7.52 RCC APB1 low-sleep clock register (RCC_APB1LLPENR) ....................................................... 462
8.7.53 RCC APB1 high-sleep clock register (RCC_APB1HLPENR) ..................................................... 465
8.7.54 RCC APB2 sleep clock register (RCC_APB2LPENR) ................................................................. 466
8.7.55 RCC APB4 sleep clock register (RCC_APB4LPENR) ................................................................. 468

8.8 RCC register map ......................................................................................................................... 470

9 Clock recovery system (CRS) ........................................................................................................... 477

9.1 Introduction ................................................................................................................................. 477
9.2 CRS main features ....................................................................................................................... 477
9.3 CRS implementation .................................................................................................................... 477
9.4 CRS functional description .......................................................................................................... 478
9.4.1 CRS block diagram .................................................................................................................. 478
9.5 CRS internal signals ................................................................................................................... 478
9.5.1 Synchronization input .............................................................................................................. 479
9.5.2 Frequency error measurement ............................................................................................... 479
9.5.3 Frequency error evaluation and automatic trimming ............................................................. 481
9.5.4 CRS initialization and configuration ....................................................................................... 481
9.6 CRS low-power modes ............................................................................................................... 482
9.7 CRS interrupts ........................................ 482
9.8 CRS registers ........................................ 483
  9.8.1 CRS control register (CRS_CR) ................. 483
  9.8.2 CRS configuration register (CRS_CFRG) .......... 484
  9.8.3 CRS interrupt and status register (CRS_ISR) .... 485
  9.8.4 CRS interrupt flag clear register (CRS_ICR) .... 487
  9.8.5 CRS register map .................................. 488

10 Hardware semaphore (HSEM) .......................... 489
  10.1 Hardware semaphore introduction .................. 489
  10.2 Hardware semaphore main features ................. 489
  10.3 HSEM functional description ....................... 490
    10.3.1 HSEM block diagram ......................... 490
    10.3.2 HSEM internal signals ....................... 490
    10.3.3 HSEM lock procedures ....................... 491
    10.3.4 HSEM write/read/readlock register address .... 492
    10.3.5 HSEM clear procedures ...................... 492
    10.3.6 HSEM MASTERID semaphore clear ............. 493
    10.3.7 HSEM interrupts ............................ 493
    10.3.8 AHB bus master ID verification ............... 495
  10.4 HSEM registers ................................... 496
    10.4.1 HSEM register semaphore x (HSEM_Rx) ........ 496
    10.4.2 HSEM read lock register semaphore x (HSEM_RLRx) 497
    10.4.3 HSEM interrupt enable register (HSEM_IER) ... 498
    10.4.4 HSEM interrupt clear register (HSEM_ICR) .... 498
    10.4.5 HSEM interrupt status register (HSEM_ISR) ... 498
    10.4.6 HSEM interrupt status register (HSEM_MISR) ... 499
    10.4.7 HSEM clear register (HSEM_CR) ................ 499
    10.4.8 HSEM interrupt clear register (HSEM_KEYR) .... 500
    10.4.9 HSEM register map ............................ 501

11 General-purpose I/Os (GPIO) .......................... 502
  11.1 Introduction ..................................... 502
  11.2 GPIO main features ................................ 502
  11.3 GPIO functional description ...................... 502
    11.3.1 General-purpose I/O (GPIO) .................. 505
11.3.2 I/O pin alternate function multiplexer and mapping ............... 505
11.3.3 I/O port control registers ........................................ 506
11.3.4 I/O port data registers ............................................. 506
11.3.5 I/O data bitwise handling .......................................... 506
11.3.6 GPIO locking mechanism ............................................. 507
11.3.7 I/O alternate function input/output .............................. 507
11.3.8 External interrupt/wakeup lines ................................... 507
11.3.9 Input configuration .................................................. 508
11.3.10 Output configuration .............................................. 508
11.3.11 I/O compensation cell ............................................. 509
11.3.12 Alternate function configuration ............................... 509
11.3.13 Analog configuration ............................................. 510
11.3.14 Using the HSE or LSE oscillator pins as GPIOs ............. 511
11.3.15 Using the GPIO pins in the backup supply domain .......... 511
11.4 GPIO registers ......................................................... 512
11.4.1 GPIO port mode register (GPIOx_MODER) (x = A to K) ......... 512
11.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to K) .... 512
11.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to K) .... 513
11.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to K) .... 513
11.4.5 GPIO port input data register (GPIOx_IDR) (x = A to K) ........... 514
11.4.6 GPIO port output data register (GPIOx_ODR) (x = A to K) ........... 514
11.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to K) ........... 514
11.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to K) ........... 515
11.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to K) ........... 516
11.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to J) ........... 517
11.4.11 GPIO register map .................................................. 519

12 System configuration controller (SYSCFG) .......................... 521
12.1 Introduction ............................................................. 521
12.2 SYSCFG main features ............................................... 521
12.3 SYSCFG functional description ........................................... 521
  12.3.1 Analog switch configuration management ......................... 521
  12.3.2 I2C Fm+ configuration .............................................. 521
  12.3.3 Timer break input lockup management ............................ 522
  12.3.4 Management of external interrupt line connections to GPIOs .... 522
  12.3.5 I/O speed in low-voltage mode .................................. 522
  12.3.6 I/O compensation cell management ................................ 522
12.4 SYSCFG registers .......................................................... 523
  12.4.1 SYSCFG peripheral mode configuration register (SYSCFG_PMCR) .. 523
  12.4.2 SYSCFG external interrupt configuration register 1
      (SYSCFG_EXTICR1) .................................................. 525
  12.4.3 SYSCFG external interrupt configuration register 2
      (SYSCFG_EXTICR2) .................................................. 526
  12.4.4 SYSCFG external interrupt configuration register 3
      (SYSCFG_EXTICR3) .................................................. 527
  12.4.5 SYSCFG external interrupt configuration register 4
      (SYSCFG_EXTICR4) .................................................. 528
  12.4.6 SYSCFG timer break lockup register
      (SYSCFG_CFGR) ..................................................... 528
  12.4.7 SYSCFG compensation cell control/status register (SYSCFG_CCCSR).
      530
  12.4.8 SYSCFG compensation cell value register (SYSCFG_CCVR) ........ 531
  12.4.9 SYSCFG compensation cell code register (SYSCFG_CCCR) ........ 532
  12.4.10 SYSCFG register maps ............................................. 533

13 Block interconnect ......................................................... 534
  13.1 Peripheral interconnect .............................................. 534
    13.1.1 Introduction .................................................... 534
    13.1.2 Connection overview ........................................... 534
  13.2 Wakeup from low-power modes ........................................ 547
  13.3 DMA ................................................................. 552
    13.3.1 MDMA (CD domain) .............................................. 553
    13.3.2 DMAMUX1, DMA1, DMA2 and BDMA1 (CD domain) ................ 555
    13.3.3 DMAMUX2, BDMA2 (SRD domain) ................................ 560

14 MDMA controller (MDMA) .................................................. 563
  14.1 MDMA introduction .................................................... 563
  14.2 MDMA main features .................................................. 563
14.3 MDMA functional description
   14.3.1 MDMA block diagram
   14.3.2 MDMA internal signals
   14.3.3 MDMA overview
   14.3.4 MDMA channel
   14.3.5 Source, destination and transfer modes
   14.3.6 Pointer update
   14.3.7 MDMA buffer transfer
   14.3.8 Request arbitration
   14.3.9 FIFO
   14.3.10 Block transfer
   14.3.11 Block repeat mode
   14.3.12 Linked list mode
   14.3.13 MDMA transfer completion
   14.3.14 MDMA transfer suspension
   14.3.15 Error management
   14.4 MDMA interrupts
   14.5 MDMA registers
      14.5.1 MDMA global interrupt/status register (MDMA_GISR0)
      14.5.2 MDMA channel x interrupt/status register (MDMA_CxISR)
      14.5.3 MDMA channel x interrupt flag clear register (MDMA_CxIFCR)
      14.5.4 MDMA channel x error status register (MDMA_CxESR)
      14.5.5 MDMA channel x control register (MDMA_CxCR)
      14.5.6 MDMA channel x transfer configuration register (MDMA_CxTCR)
      14.5.7 MDMA channel x block number of data register (MDMA_CxBNDTR)
      14.5.8 MDMA channel x source address register (MDMA_CxSAR)
      14.5.9 MDMA channel x destination address register (MDMA_CxDAR)
      14.5.10 MDMA channel x block repeat address update register (MDMA_CxBRUR)
      14.5.11 MDMA channel x link address register (MDMA_CxLAR)
      14.5.12 MDMA channel x trigger and bus selection register (MDMA_CxTBR)
      14.5.13 MDMA channel x mask address register (MDMA_CxMAR)
      14.5.14 MDMA channel x mask data register (MDMA_CxMDR)
      14.5.15 MDMA register map

15 Direct memory access controller (DMA)
   15.1 DMA introduction
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.2</td>
<td>DMA main features ........................................ 592</td>
</tr>
<tr>
<td>15.3</td>
<td>DMA functional description .............................. 594</td>
</tr>
<tr>
<td>15.3.1</td>
<td>DMA block diagram ........................................ 594</td>
</tr>
<tr>
<td>15.3.2</td>
<td>DMA internal signals ...................................... 594</td>
</tr>
<tr>
<td>15.3.3</td>
<td>DMA overview ............................................... 594</td>
</tr>
<tr>
<td>15.3.4</td>
<td>DMA transactions ........................................... 595</td>
</tr>
<tr>
<td>15.3.5</td>
<td>DMA request mapping ....................................... 595</td>
</tr>
<tr>
<td>15.3.6</td>
<td>Arbiter ...................................................... 596</td>
</tr>
<tr>
<td>15.3.7</td>
<td>DMA streams ................................................ 596</td>
</tr>
<tr>
<td>15.3.8</td>
<td>Source, destination and transfer modes ................. 596</td>
</tr>
<tr>
<td>15.3.9</td>
<td>Pointer incrementation .................................... 600</td>
</tr>
<tr>
<td>15.3.10</td>
<td>Circular mode .............................................. 601</td>
</tr>
<tr>
<td>15.3.11</td>
<td>Double-buffer mode ....................................... 601</td>
</tr>
<tr>
<td>15.3.12</td>
<td>Programmable data width, packing/unpacking, endianness .. 602</td>
</tr>
<tr>
<td>15.3.13</td>
<td>Single and burst transfers ................................ 604</td>
</tr>
<tr>
<td>15.3.14</td>
<td>FIFO .......................................................... 604</td>
</tr>
<tr>
<td>15.3.15</td>
<td>DMA transfer completion ................................... 607</td>
</tr>
<tr>
<td>15.3.16</td>
<td>DMA transfer suspension .................................... 608</td>
</tr>
<tr>
<td>15.3.17</td>
<td>Flow controller ............................................. 609</td>
</tr>
<tr>
<td>15.3.18</td>
<td>Summary of the possible DMA configurations ............. 610</td>
</tr>
<tr>
<td>15.3.19</td>
<td>Stream configuration procedure ........................ 610</td>
</tr>
<tr>
<td>15.3.20</td>
<td>Error management .......................................... 611</td>
</tr>
<tr>
<td>15.4</td>
<td>DMA interrupts ............................................... 612</td>
</tr>
<tr>
<td>15.5</td>
<td>DMA registers ................................................ 613</td>
</tr>
<tr>
<td>15.5.1</td>
<td>DMA low interrupt status register (DMA_LISR) .......... 613</td>
</tr>
<tr>
<td>15.5.2</td>
<td>DMA high interrupt status register (DMA_HISR) ........ 614</td>
</tr>
<tr>
<td>15.5.3</td>
<td>DMA low interrupt flag clear register (DMA_LIFCR) ..... 615</td>
</tr>
<tr>
<td>15.5.4</td>
<td>DMA high interrupt flag clear register (DMA_HIFCR) .... 615</td>
</tr>
<tr>
<td>15.5.5</td>
<td>DMA stream x configuration register (DMA_SxCR) ........ 616</td>
</tr>
<tr>
<td>15.5.6</td>
<td>DMA stream x number of data register (DMA_SxDTR) .... 619</td>
</tr>
<tr>
<td>15.5.7</td>
<td>DMA stream x peripheral address register (DMA_SxPAR) .. 620</td>
</tr>
<tr>
<td>15.5.8</td>
<td>DMA stream x memory 0 address register (DMA_SxM0AR) ... 620</td>
</tr>
<tr>
<td>15.5.9</td>
<td>DMA stream x memory 1 address register (DMA_SxM1AR) ... 621</td>
</tr>
<tr>
<td>15.5.10</td>
<td>DMA stream x FIFO control register (DMA_SxFCR) ........ 621</td>
</tr>
<tr>
<td>15.5.11</td>
<td>DMA register map .......................................... 623</td>
</tr>
</tbody>
</table>
16 Basic direct memory access controller (BDMA) .......................... 627
  16.1 Introduction ......................................................... 627
  16.2 BDMA main features ............................................... 627
  16.3 BDMA implementation ............................................. 628
    16.3.1 BDMA1 and BDMA2 ........................................... 628
    16.3.2 BDMA request mapping ...................................... 628
  16.4 BDMA functional description .................................... 629
    16.4.1 BDMA block diagram .......................................... 629
    16.4.2 BDMA pins and internal signals ............................... 631
    16.4.3 BDMA transfers ................................................. 631
    16.4.4 BDMA arbitration .............................................. 632
    16.4.5 BDMA channels ................................................ 632
    16.4.6 BDMA data width, alignment and endianness ................ 637
    16.4.7 BDMA error management ...................................... 638
  16.5 BDMA interrupts .................................................. 639
  16.6 BDMA registers ..................................................... 639
    16.6.1 BDMA interrupt status register (BDMA_ISR) ................. 639
    16.6.2 BDMA interrupt flag clear register (BDMA_IFCR) .......... 642
    16.6.3 BDMA channel x configuration register (BDMA_CCRx) ....... 643
    16.6.4 BDMA channel x number of data to transfer register
      (BDMA_CNDTRx) .................................................... 647
    16.6.5 BDMA channel x peripheral address register (BDMA_CPARx) . 647
    16.6.6 BDMA channel x memory 0 address register (BDMA_CM0ARx) . 648
    16.6.7 BDMA channel x memory 1 address register (BDMA_CM1ARx) . 649
    16.6.8 BDMA register map ............................................. 649

17 DMA request multiplexer (DMAMUX) ................................. 652
  17.1 Introduction ........................................................ 652
  17.2 DMAMUX main features ............................................ 653
  17.3 DMAMUX implementation .......................................... 653
    17.3.1 DMAMUX1 and DMAMUX2 instantiation ......................... 653
    17.3.2 DMAMUX1 mapping ............................................. 653
    17.3.3 DMAMUX2 mapping ............................................. 655
  17.4 DMAMUX functional description .................................. 658
    17.4.1 DMAMUX block diagram ....................................... 658
    17.4.2 DMAMUX signals ............................................... 659
17.4.3 DMAMUX channels ........................................... 659
17.4.4 DMAMUX request line multiplexer .......................... 659
17.4.5 DMAMUX request generator ................................. 662
17.5 DMAMUX interrupts ............................................. 663
17.6 DMAMUX registers .............................................. 664
   17.6.1 DMAMUX1 request line multiplexer channel \(x\) configuration register (DMAMUX1_CxCR) .................................................................................. 664
   17.6.2 DMAMUX2 request line multiplexer channel \(x\) configuration register (DMAMUX2_CxCR) .................................................................................. 665
   17.6.3 DMAMUX1 request line multiplexer interrupt channel status register (DMAMUX1_CSR) ................................................................. 666
   17.6.4 DMAMUX2 request line multiplexer interrupt channel status register (DMAMUX2_CSR) ................................................................. 666
   17.6.5 DMAMUX1 request line multiplexer interrupt clear flag register (DMAMUX1_CFR) ............................................................................. 667
   17.6.6 DMAMUX2 request line multiplexer interrupt clear flag register (DMAMUX2_CFR) ............................................................................. 667
   17.6.7 DMAMUX1 request generator channel \(x\) configuration register (DMAMUX1_RGxCR) ................................................................. 668
   17.6.8 DMAMUX2 request generator channel \(x\) configuration register (DMAMUX2_RGxCR) ................................................................. 668
   17.6.9 DMAMUX1 request generator interrupt status register (DMAMUX1_RGSR) ................................................................. 670
   17.6.10 DMAMUX2 request generator interrupt status register (DMAMUX2_RGSR) ............................................................... 670
   17.6.11 DMAMUX1 request generator interrupt clear flag register (DMAMUX1_RGCFR) ............................................................... 671
   17.6.12 DMAMUX2 request generator interrupt clear flag register (DMAMUX2_RGCFR) ............................................................... 671
   17.6.13 DMAMUX register map ........................................... 672

18 Chrom-Art Accelerator™ controller (DMA2D) .................. 674
   18.1 DMA2D introduction ............................................. 674
   18.2 DMA2D main features ............................................. 674
   18.3 DMA2D functional description ............................... 675
      18.3.1 General description ........................................... 675
   18.4 DMA2D pins and internal signals ............................. 676
      18.4.1 DMA2D control ............................................. 677
      18.4.2 DMA2D foreground and background FIFOs ........... 677
      18.4.3 DMA2D foreground and background pixel format converter (PFC) .... 677
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.4.4 DMA2D foreground and background CLUT interface</td>
<td>680</td>
</tr>
<tr>
<td>18.4.5 DMA2D blender</td>
<td>681</td>
</tr>
<tr>
<td>18.4.6 DMA2D output PFC</td>
<td>681</td>
</tr>
<tr>
<td>18.4.7 DMA2D output FIFO</td>
<td>682</td>
</tr>
<tr>
<td>18.4.8 DMA2D output FIFO byte reordering</td>
<td>682</td>
</tr>
<tr>
<td>18.4.9 DMA2D AXI master port timer</td>
<td>684</td>
</tr>
<tr>
<td>18.4.10 DMA2D transactions</td>
<td>684</td>
</tr>
<tr>
<td>18.4.11 DMA2D configuration</td>
<td>685</td>
</tr>
<tr>
<td>18.4.12 YCbCr support</td>
<td>689</td>
</tr>
<tr>
<td>18.4.13 DMA2D transfer control (start, suspend, abort and completion)</td>
<td>689</td>
</tr>
<tr>
<td>18.4.14 Watermark</td>
<td>689</td>
</tr>
<tr>
<td>18.4.15 Error management</td>
<td>690</td>
</tr>
<tr>
<td>18.4.16 AXI dead time</td>
<td>690</td>
</tr>
<tr>
<td>18.5 DMA2D interrupts</td>
<td>690</td>
</tr>
<tr>
<td>18.6 DMA2D registers</td>
<td>691</td>
</tr>
<tr>
<td>18.6.1 DMA2D control register (DMA2D_CR)</td>
<td>691</td>
</tr>
<tr>
<td>18.6.2 DMA2D interrupt status register (DMA2D_ISR)</td>
<td>693</td>
</tr>
<tr>
<td>18.6.3 DMA2D interrupt flag clear register (DMA2D_IFCR)</td>
<td>694</td>
</tr>
<tr>
<td>18.6.4 DMA2D foreground memory address register (DMA2D_FGMAR)</td>
<td>695</td>
</tr>
<tr>
<td>18.6.5 DMA2D foreground offset register (DMA2D_FGOR)</td>
<td>695</td>
</tr>
<tr>
<td>18.6.6 DMA2D background memory address register (DMA2D_BGMAR)</td>
<td>696</td>
</tr>
<tr>
<td>18.6.7 DMA2D background offset register (DMA2D_BGOR)</td>
<td>696</td>
</tr>
<tr>
<td>18.6.8 DMA2D foreground PFC control register (DMA2D_FGPFCCR)</td>
<td>697</td>
</tr>
<tr>
<td>18.6.9 DMA2D foreground color register (DMA2D_FGCOLR)</td>
<td>699</td>
</tr>
<tr>
<td>18.6.10 DMA2D background PFC control register (DMA2D_BGPFCCR)</td>
<td>700</td>
</tr>
<tr>
<td>18.6.11 DMA2D background color register (DMA2D_BGCOLR)</td>
<td>702</td>
</tr>
<tr>
<td>18.6.12 DMA2D foreground CLUT memory address register (DMA2D_FGCMAR)</td>
<td>702</td>
</tr>
<tr>
<td>18.6.13 DMA2D background CLUT memory address register (DMA2D_BGCMAR)</td>
<td>703</td>
</tr>
<tr>
<td>18.6.14 DMA2D output PFC control register (DMA2D_OPFCCR)</td>
<td>703</td>
</tr>
<tr>
<td>18.6.15 DMA2D output color register (DMA2D_OCOLR)</td>
<td>704</td>
</tr>
<tr>
<td>18.6.16 DMA2D output memory address register (DMA2D_OMAR)</td>
<td>706</td>
</tr>
<tr>
<td>18.6.17 DMA2D output offset register (DMA2D_OOR)</td>
<td>706</td>
</tr>
<tr>
<td>18.6.18 DMA2D number of line register (DMA2D_NLR)</td>
<td>707</td>
</tr>
<tr>
<td>18.6.19 DMA2D line watermark register (DMA2D_LWR)</td>
<td>707</td>
</tr>
<tr>
<td>18.6.20 DMA2D AXI master timer configuration register (DMA2D_AMTCR)</td>
<td>708</td>
</tr>
</tbody>
</table>
18.6.21 DMA2D foreground CLUT (DMA2D_FGCLUT[y]) .......................... 708
18.6.22 DMA2D background CLUT (DMA2D_BGCLUT[y]) .................. 709
18.6.23 DMA2D register map ............................................... 710

19 Nested vectored interrupt controller (NVIC) ......................... 712
19.1 NVIC features ..................................................................... 712
  19.1.1 SysTick calibration value register .................................. 712
  19.1.2 Interrupt and exception vectors .................................. 712

20 Extended interrupt and event controller (EXTI) ................... 721
20.1 EXTI main features .......................................................... 721
20.2 EXTI block diagram .......................................................... 722
  20.2.1 EXTI connections between peripherals, CPU and SRD domain .. 722
20.3 EXTI functional description ................................................. 723
  20.3.1 EXTI configurable event input CPU wakeup ..................... 724
  20.3.2 EXTI configurable event input Any wakeup ...................... 725
  20.3.3 EXTI direct event input CPU wakeup ............................. 727
  20.3.4 EXTI direct event input Any wakeup ............................. 728
  20.3.5 EXTI SRD pending request clear selection ...................... 729
20.4 EXTI event input mapping .................................................. 729
20.5 EXTI functional behavior .................................................... 732
  20.5.1 EXTI CPU interrupt procedure ...................................... 733
  20.5.2 EXTI CPU event procedure .......................................... 733
  20.5.3 EXTI CPU wakeup procedure ....................................... 734
  20.5.4 EXTI SRD domain wakeup for Autonomous run mode procedure 734
  20.5.5 EXTI software interrupt/event trigger procedure .............. 734
20.6 EXTI registers ................................................................. 735
  20.6.1 EXTI rising trigger selection register (EXTI_RTSR1) .......... 735
  20.6.2 EXTI falling trigger selection register (EXTI_FTSR1) ........... 735
  20.6.3 EXTI software interrupt event register (EXTI_SWIER1) ....... 736
  20.6.4 EXTI SRD pending mask register (EXTI_SRDPMR1) .......... 736
  20.6.5 EXTI SRD pending clear selection register low (EXTI_SRDPCR1L) 737
  20.6.6 EXTI SRD pending clear selection register high (EXTI_SRDPCR1H) 738
  20.6.7 EXTI rising trigger selection register (EXTI_RTSR2) .......... 738
  20.6.8 EXTI falling trigger selection register (EXTI_FTSR2) .......... 739
  20.6.9 EXTI software interrupt event register (EXTI_SWIER2) ...... 740
20.6.10 EXTI SRD pending mask register (EXTI_SRDPMR2) .................................. 741
20.6.11 EXTI SRD pending clear selection register low (EXTI_SRDP CLR2L) . 741
20.6.12 EXTI SRD pending clear selection register high (EXTI_SRDP CLR2H) 742
20.6.13 EXTI rising trigger selection register (EXTI_RTSR3) .............................. 743
20.6.14 EXTI falling trigger selection register (EXTI_FTSR3) .................................. 743
20.6.15 EXTI software interrupt event register (EXTI_S WIER3) ......................... 744
20.6.16 EXTI SRD pending mask register (EXTI_SRDP MR3) ................................. 745
20.6.17 EXTI SRD pending clear selection register high (EXTI_SRDP CLR3H) 745
20.6.18 EXTI interrupt mask register (EXTI_C PUIMR1) ....................................... 746
20.6.19 EXTI event mask register (EXTI_C PUEMR1) ............................................. 746
20.6.20 EXTI pending register (EXTI_ CPUPR1) .................................................. 747
20.6.21 EXTI interrupt mask register (EXTI_C PUIMR2) ....................................... 747
20.6.22 EXTI event mask register (EXTI_C PUEMR2) ............................................. 748
20.6.23 EXTI pending register (EXTI_ CPUPR2) .................................................. 749
20.6.24 EXTI interrupt mask register (EXTI_C PUIMR3) ....................................... 749
20.6.25 EXTI event mask register (EXTI_C PUEMR3) ............................................. 751
20.6.26 EXTI pending register (EXTI_ CPUPR3) .................................................. 752
20.6.27 EXTI register map ..................................................................................... 752

21 Chrom-GRC™ (GFXMMU) ................................................................................ 755
21.1 Introduction .................................................................................................... 755
21.2 Chrom-GRC™ main features ....................................................................... 755
21.3 Chrom-GRC™ functional and architectural description ............................... 756
  21.3.1 Virtual memory ......................................................................................... 756
  21.3.2 MMU architecture .................................................................................... 758
  21.3.3 Cache and prefetch mechanism .............................................................. 761
21.4 Graphic MMU interrupts .............................................................................. 764
21.5 Graphic MMU registers .............................................................................. 765
  21.5.1 Graphic MMU configuration register (GFXMMU_C R) ......................... 765
  21.5.2 Graphic MMU status register (GFXMMU_SR) ........................................ 766
  21.5.3 Graphic MMU flag clear register (GFXMMU_F CR) .............................. 767
  21.5.4 Graphic MMU cache control register (GFXMMU_C CR) .................... 767
  21.5.5 Graphic MMU default value register (GFXMMU_D VR) ....................... 768
  21.5.6 Graphic MMU buffer 0 configuration register (GFXMMU_B0 CR) .... 768
  21.5.7 Graphic MMU buffer 1 configuration register (GFXMMU_B1 CR) ....... 769
  21.5.8 Graphic MMU buffer 2 configuration register (GFXMMU_B2 CR) ....... 769
  21.5.9 Graphic MMU buffer 3 configuration register (GFXMMU_B3 CR) ....... 770
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.5.10</td>
<td>Graphic MMU LUT entry x low (GFXMMU_LUTxL)</td>
<td>770</td>
</tr>
<tr>
<td>21.5.11</td>
<td>Graphic MMU LUT entry x high (GFXMMU_LUTxH)</td>
<td>770</td>
</tr>
<tr>
<td>21.5.12</td>
<td>Graphic MMU register map</td>
<td>772</td>
</tr>
<tr>
<td>22</td>
<td>Cyclic redundancy check calculation unit (CRC)</td>
<td>773</td>
</tr>
<tr>
<td>22.1</td>
<td>Introduction</td>
<td>773</td>
</tr>
<tr>
<td>22.2</td>
<td>CRC main features</td>
<td>773</td>
</tr>
<tr>
<td>22.3</td>
<td>CRC functional description</td>
<td>774</td>
</tr>
<tr>
<td>22.3.1</td>
<td>CRC block diagram</td>
<td>774</td>
</tr>
<tr>
<td>22.3.2</td>
<td>CRC internal signals</td>
<td>774</td>
</tr>
<tr>
<td>22.3.3</td>
<td>CRC operation</td>
<td>774</td>
</tr>
<tr>
<td>22.4</td>
<td>CRC registers</td>
<td>776</td>
</tr>
<tr>
<td>22.4.1</td>
<td>CRC data register (CRC_DR)</td>
<td>776</td>
</tr>
<tr>
<td>22.4.2</td>
<td>CRC independent data register (CRC_IDR)</td>
<td>776</td>
</tr>
<tr>
<td>22.4.3</td>
<td>CRC control register (CRC_CR)</td>
<td>777</td>
</tr>
<tr>
<td>22.4.4</td>
<td>CRC initial value (CRC_INIT)</td>
<td>777</td>
</tr>
<tr>
<td>22.4.5</td>
<td>CRC polynomial (CRC_POL)</td>
<td>778</td>
</tr>
<tr>
<td>22.4.6</td>
<td>CRC register map</td>
<td>778</td>
</tr>
<tr>
<td>23</td>
<td>Flexible memory controller (FMC)</td>
<td>779</td>
</tr>
<tr>
<td>23.1</td>
<td>FMC main features</td>
<td>779</td>
</tr>
<tr>
<td>23.2</td>
<td>FMC block diagram</td>
<td>780</td>
</tr>
<tr>
<td>23.3</td>
<td>FMC internal signals</td>
<td>782</td>
</tr>
<tr>
<td>23.4</td>
<td>AHB interface</td>
<td>782</td>
</tr>
<tr>
<td>23.5</td>
<td>AXI interface</td>
<td>782</td>
</tr>
<tr>
<td>23.5.1</td>
<td>Supported memories and transactions</td>
<td>783</td>
</tr>
<tr>
<td>23.6</td>
<td>External device address mapping</td>
<td>784</td>
</tr>
<tr>
<td>23.6.1</td>
<td>NOR/PSRAM address mapping</td>
<td>785</td>
</tr>
<tr>
<td>23.6.2</td>
<td>NAND Flash memory address mapping</td>
<td>786</td>
</tr>
<tr>
<td>23.6.3</td>
<td>SDRAM address mapping</td>
<td>787</td>
</tr>
<tr>
<td>23.7</td>
<td>NOR Flash/PSRAM controller</td>
<td>790</td>
</tr>
<tr>
<td>23.7.1</td>
<td>External memory interface signals</td>
<td>792</td>
</tr>
<tr>
<td>23.7.2</td>
<td>Supported memories and transactions</td>
<td>794</td>
</tr>
<tr>
<td>23.7.3</td>
<td>General timing rules</td>
<td>795</td>
</tr>
<tr>
<td>23.7.4</td>
<td>NOR Flash/PSRAM controller asynchronous transactions</td>
<td>795</td>
</tr>
<tr>
<td>23.7.5</td>
<td>Synchronous transactions</td>
<td>815</td>
</tr>
</tbody>
</table>
23.7.6 NOR/PSRAM controller registers ........................................ 821
23.8 NAND Flash controller ....................................................... 830
  23.8.1 External memory interface signals .................................. 830
  23.8.2 NAND Flash supported memories and transactions ............... 831
  23.8.3 Timing diagrams for NAND Flash memories ......................... 832
  23.8.4 NAND Flash operations ............................................... 833
  23.8.5 NAND Flash prewait feature ........................................ 834
  23.8.6 Computation of the error correction code (ECC) in NAND Flash memory ........................................ 835
  23.8.7 NAND Flash controller registers .................................... 836
23.9 SDRAM controller ............................................................ 842
  23.9.1 SDRAM controller main features ..................................... 842
  23.9.2 SDRAM External memory interface signals ......................... 842
  23.9.3 SDRAM controller functional description ......................... 843
  23.9.4 Low-power modes ..................................................... 850
  23.9.5 SDRAM controller registers ......................................... 853
  23.9.6 FMC register map .................................................... 859
24 Octo-SPI interface (OCTOSPI) ............................................. 862
  24.1 Introduction .............................................................. 862
  24.2 OCTOSPI main features .................................................. 862
  24.3 OCTOSPI functional description ....................................... 863
    24.3.1 OCTOSPI block diagram ........................................... 863
    24.3.2 OCTOSPI interface to memory modes ............................. 864
    24.3.3 OCTOSPI Regular-command mode .................................. 864
    24.3.4 OCTOSPI Regular-command mode signal interface ............. 868
    24.3.5 HyperBus mode ................................................... 871
    24.3.6 Common functionality between the Regular-command and HyperBus modes ........................................ 875
    24.3.7 OCTOSPI operating modes introduction .......................... 877
    24.3.8 OCTOSPI Indirect mode ............................................ 877
    24.3.9 OCTOSPI Status-flag polling mode ................................ 879
    24.3.10 OCTOSPI Memory-mapped mode ................................... 879
    24.3.11 OCTOSPI configuration introduction ............................ 880
    24.3.12 OCTOSPI system configuration ................................... 880
    24.3.13 OCTOSPI device configuration ................................... 881
    24.3.14 OCTOSPI Regular-command mode configuration ................. 881
24.3.15 OCTOSPI HyperBus mode configuration .......................... 884
24.3.16 OCTOSPI error management ................................. 886
24.3.17 OCTOSPI busy bit and abort functionality ................. 886
24.3.18 OCTOSPI reconfiguration or deactivation .................. 887
24.3.19 nCS behavior .................................................. 887
24.4 OCTOSPI interrupts ................................................. 888
24.5 OCTOSPI registers .................................................. 889
  24.5.1 OCTOSPI control register (OCTOSPI_CR) ...................... 889
  24.5.2 OCTOSPI device configuration register 1 (OCTOSPI_DCR1) ...... 892
  24.5.3 OCTOSPI device configuration register 2 (OCTOSPI_DCR2) .... 893
  24.5.4 OCTOSPI device configuration register 3 (OCTOSPI_DCR3) .... 894
  24.5.5 OCTOSPI device configuration register 4 (OCTOSPI_DCR4) .... 895
  24.5.6 OCTOSPI status register (OCTOSPI_SR) ..................... 896
  24.5.7 OCTOSPI flag clear register (OCTOSPI_FCR) ................. 897
  24.5.8 OCTOSPI data length register (OCTOSPI_DLR) .............. 897
  24.5.9 OCTOSPI address register (OCTOSPI_AR) .................. 898
  24.5.10 OCTOSPI data register (OCTOSPI_DR) .................... 898
  24.5.11 OCTOSPI polling status mask register (OCTOSPI_PSMKR) .... 899
  24.5.12 OCTOSPI polling status match register (OCTOSPI_PSMAR) ... 900
  24.5.13 OCTOSPI polling interval register (OCTOSPI_PIR) ......... 900
  24.5.14 OCTOSPI communication configuration register (OCTOSPI_CCR) . 900
  24.5.15 OCTOSPI timing configuration register (OCTOSPI_TCR) ....... 903
  24.5.16 OCTOSPI instruction register (OCTOSPI_IR) ............... 903
  24.5.17 OCTOSPI alternate bytes register (OCTOSPI_ABR) .......... 904
  24.5.18 OCTOSPI low-power timeout register (OCTOSPI_LPTR) ....... 904
  24.5.19 OCTOSPI wrap communication configuration register (OCTOSPI_WPCCR) . 905
  24.5.20 OCTOSPI wrap timing configuration register (OCTOSPI_WPTCR) ..... 907
  24.5.21 OCTOSPI wrap instruction register (OCTOSPI_WPIR) ....... 908
  24.5.22 OCTOSPI wrap alternate bytes register (OCTOSPI_WPABR) ... 908
  24.5.23 OCTOSPI write communication configuration register (OCTOSPI_WCCR) . 909
  24.5.24 OCTOSPI write timing configuration register (OCTOSPI_WTCR) ... 911
  24.5.25 OCTOSPI write instruction register (OCTOSPI_WIR) ....... 912
  24.5.26 OCTOSPI write alternate bytes register (OCTOSPI_WABR) ... 912
  24.5.27 OCTOSPI HyperBus latency configuration register (OCTOSPI_HLCR) .... 912
  24.5.28 OCTOSPI register map ...................................... 913
<table>
<thead>
<tr>
<th>Contents</th>
<th>RM0455</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>25</strong> OCTOSPI I/O manager (OCTOSPIM)</td>
<td>917</td>
</tr>
<tr>
<td>25.1 Introduction</td>
<td>917</td>
</tr>
<tr>
<td>25.2 OCTOSPIM main features</td>
<td>917</td>
</tr>
<tr>
<td>25.3 OCTOSPIM functional description</td>
<td>917</td>
</tr>
<tr>
<td>25.3.1 OCTOSPIM block diagram</td>
<td>917</td>
</tr>
<tr>
<td>25.3.2 OCTOSPIM matrix</td>
<td>918</td>
</tr>
<tr>
<td>25.3.3 OCTOSPIM multiplexer</td>
<td>919</td>
</tr>
<tr>
<td>25.4 OCTOSPI I/O manager registers</td>
<td>920</td>
</tr>
<tr>
<td>25.4.1 OCTOSPI I/O manager control register</td>
<td>920</td>
</tr>
<tr>
<td>25.4.2 OCTOSPI I/O manager Port n configuration register</td>
<td>920</td>
</tr>
<tr>
<td>25.4.3 OCTOSPIM register map</td>
<td>923</td>
</tr>
<tr>
<td><strong>26</strong> Delay block (DLYB)</td>
<td>924</td>
</tr>
<tr>
<td>26.1 Introduction</td>
<td>924</td>
</tr>
<tr>
<td>26.2 DLYB main features</td>
<td>924</td>
</tr>
<tr>
<td>26.3 DLYB functional description</td>
<td>924</td>
</tr>
<tr>
<td>26.3.1 DLYB diagram</td>
<td>924</td>
</tr>
<tr>
<td>26.3.2 DLYB pins and internal signals</td>
<td>925</td>
</tr>
<tr>
<td>26.3.3 General description</td>
<td>925</td>
</tr>
<tr>
<td>26.3.4 Delay line length configuration procedure</td>
<td>926</td>
</tr>
<tr>
<td>26.3.5 Output clock phase configuration procedure</td>
<td>926</td>
</tr>
<tr>
<td>26.4 DLYB registers</td>
<td>927</td>
</tr>
<tr>
<td>26.4.1 DLYB control register (DLYB_CR)</td>
<td>927</td>
</tr>
<tr>
<td>26.4.2 DLYB configuration register (DLYB_CFGR)</td>
<td>927</td>
</tr>
<tr>
<td>26.4.3 DLYB register map</td>
<td>928</td>
</tr>
<tr>
<td><strong>27</strong> Analog-to-digital converters (ADC)</td>
<td>929</td>
</tr>
<tr>
<td>27.1 Introduction</td>
<td>929</td>
</tr>
<tr>
<td>27.2 ADC main features</td>
<td>930</td>
</tr>
<tr>
<td>27.3 ADC implementation</td>
<td>931</td>
</tr>
<tr>
<td>27.4 ADC functional description</td>
<td>932</td>
</tr>
<tr>
<td>27.4.1 ADC block diagram</td>
<td>932</td>
</tr>
<tr>
<td>27.4.2 ADC pins and internal signals</td>
<td>933</td>
</tr>
<tr>
<td>27.4.3 ADC clocks</td>
<td>934</td>
</tr>
<tr>
<td>27.4.4 ADC1/2 connectivity</td>
<td>936</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>27.4.5</td>
<td>Slave AHB interface</td>
</tr>
<tr>
<td>27.4.6</td>
<td>ADC deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)</td>
</tr>
<tr>
<td>27.4.7</td>
<td>Single-ended and differential input channels</td>
</tr>
<tr>
<td>27.4.8</td>
<td>Calibration (ADCAL, ADCALDIFF, ADCALLIN, ADC_CALFACT)</td>
</tr>
<tr>
<td>27.4.9</td>
<td>ADC on-off control (ADEN, ADDIS, ADRDY)</td>
</tr>
<tr>
<td>27.4.10</td>
<td>Constraints when writing the ADC control bits</td>
</tr>
<tr>
<td>27.4.11</td>
<td>Channel selection (SQRx, JSQRx)</td>
</tr>
<tr>
<td>27.4.12</td>
<td>Channel preselection register (ADC_PCSEL)</td>
</tr>
<tr>
<td>27.4.13</td>
<td>Channel-wise programmable sampling time (SMPR1, SMPR2)</td>
</tr>
<tr>
<td>27.4.14</td>
<td>Single conversion mode (CONT=0)</td>
</tr>
<tr>
<td>27.4.15</td>
<td>Continuous conversion mode (CONT=1)</td>
</tr>
<tr>
<td>27.4.16</td>
<td>Starting conversions (ADSTART, JADSTART)</td>
</tr>
<tr>
<td>27.4.17</td>
<td>Timing</td>
</tr>
<tr>
<td>27.4.18</td>
<td>Stopping an ongoing conversion (ADSTP, JADSTP)</td>
</tr>
<tr>
<td>27.4.19</td>
<td>Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)</td>
</tr>
<tr>
<td>27.4.20</td>
<td>Injected channel management</td>
</tr>
<tr>
<td>27.4.21</td>
<td>Discontinuous mode (DISCEN, DISCNUM, JDISCEN)</td>
</tr>
<tr>
<td>27.4.22</td>
<td>Queue of context for injected conversions</td>
</tr>
<tr>
<td>27.4.23</td>
<td>Programmable resolution (RES) - fast conversion mode</td>
</tr>
<tr>
<td>27.4.24</td>
<td>End of conversion, end of sampling phase (EOC, JEOC, EOSMP)</td>
</tr>
<tr>
<td>27.4.25</td>
<td>End of conversion sequence (EOS, JEOS)</td>
</tr>
<tr>
<td>27.4.26</td>
<td>Timing diagrams example (single/continuous modes, hardware/software triggers)</td>
</tr>
<tr>
<td>27.4.27</td>
<td>Data management</td>
</tr>
<tr>
<td>27.4.28</td>
<td>Managing conversions using the DFSDM</td>
</tr>
<tr>
<td>27.4.29</td>
<td>Dynamic low-power features</td>
</tr>
<tr>
<td>27.4.30</td>
<td>Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)</td>
</tr>
<tr>
<td>27.4.31</td>
<td>Oversampler</td>
</tr>
<tr>
<td>27.4.32</td>
<td>Dual ADC modes</td>
</tr>
<tr>
<td>27.4.33</td>
<td>Temperature sensor</td>
</tr>
<tr>
<td>27.4.34</td>
<td>VBAT supply monitoring</td>
</tr>
<tr>
<td>27.4.35</td>
<td>Monitoring the internal voltage reference</td>
</tr>
<tr>
<td>27.5</td>
<td>ADC interrupts</td>
</tr>
<tr>
<td>27.6</td>
<td>ADC registers (for each ADC)</td>
</tr>
<tr>
<td>27.6.1</td>
<td>ADC interrupt and status register (ADC_ISR)</td>
</tr>
</tbody>
</table>
27.6.2 ADC interrupt enable register (ADC_IER) .......................... 1014
27.6.3 ADC control register (ADC_CR) ................................. 1016
27.6.4 ADC configuration register (ADC_CFGR) ....................... 1021
27.6.5 ADC configuration register 2 (ADC_CFGR2) ...................... 1025
27.6.6 ADC sample time register 1 (ADC_SMPR1) ..................... 1027
27.6.7 ADC sample time register 2 (ADC_SMPR2) ..................... 1028
27.6.8 ADC channel preselection register (ADC_PCSEL) ............. 1029
27.6.9 ADC watchdog threshold register 1 (ADC_LTR1) ............. 1029
27.6.10 ADC watchdog threshold register 1 (ADC_HTR1) ........... 1030
27.6.11 ADC regular sequence register 1 (ADC_SQR1) .............. 1031
27.6.12 ADC regular sequence register 2 (ADC_SQR2) .............. 1032
27.6.13 ADC regular sequence register 3 (ADC_SQR3) .............. 1033
27.6.14 ADC regular sequence register 4 (ADC_SQR4) .............. 1034
27.6.15 ADC regular Data Register (ADC_DR) ...................... 1035
27.6.16 ADC injected sequence register (ADC_JSQR) ............... 1036
27.6.17 ADC injected channel y offset register (ADC_OFRy) ....... 1038
27.6.18 ADC injected channel y data register (ADC_JDRy) ......... 1039
27.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR) .................. 1039
27.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR) .................. 1040
27.6.21 ADC watchdog lower threshold register 2 (ADC_LTR2) .... 1040
27.6.22 ADC watchdog higher threshold register 2 (ADC_HTR2) ... 1041
27.6.23 ADC watchdog lower threshold register 3 (ADC_LTR3) .... 1041
27.6.24 ADC watchdog higher threshold register 3 (ADC_HTR3) ... 1042
27.6.25 ADC differential mode selection register (ADC_DIFSEL) ... 1042
27.6.26 ADC calibration factors register (ADC_CALFACT) ......... 1043
27.6.27 ADC calibration factor register 2 (ADC_CALFACT2) ... 1043

27.7 ADC common registers ................................................. 1044
27.7.1 ADC xcommon status register (ADCx_CSR) (x=1/2) .......... 1044
27.7.2 ADC x common control register (ADCx_CCR) (x=1/2) ....... 1046
27.7.3 ADC x common regular data register for dual mode (ADCx_CDR) (x=1/2) .... 1049
27.7.4 ADC x common regular data register for 32-bit dual mode (ADCx_CDR2) (x=1/2) .... 1049

27.8 ADC register map ..................................................... 1050

28 Digital temperature sensor (DTS) .................................... 1054
## 28. DTS (Digital-to-Temperature Sensor)

**28.1 Introduction** ................................. 1054

**28.2 DTS main features** ...................... 1054

**28.3 DTS functional description** ............. 1055

- **28.3.1 DTS block diagram** ..................... 1055
- **28.3.2 DTS internal signals** .................. 1055
- **28.3.3 DTS block operation** ................... 1056
- **28.3.4 Operating modes** ....................... 1056
- **28.3.5 Calibration** ............................ 1056
- **28.3.6 Prescaler** .............................. 1056
- **28.3.7 Temperature measurement principles** .... 1057
- **28.3.8 Sampling time** ......................... 1058
- **28.3.9 Trigger input** .......................... 1059
- **28.3.10 On-off control and ready flag** ........ 1059
- **28.3.11 Temperature measurement sequence** .... 1060

**28.4 DTS low-power modes** .................... 1061

**28.5 DTS interrupts** ............................ 1061

- **28.5.1 Temperature window comparator** ........ 1061
- **28.5.2 Synchronous interrupt** .................. 1061
- **28.5.3 Asynchronous wakeup** ................... 1061

**28.6 DTS registers** ............................. 1063

- **28.6.1 Temperature sensor configuration register 1 (DTS_CFRG1)** .... 1063
- **28.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1)** ......... 1064
- **28.6.3 Temperature sensor ramp value register (DTS_RAMPVALR)** ......... 1064
- **28.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1)** .... 1065
- **28.6.5 Temperature sensor data register (DTS_DR)** .................... 1065
- **28.6.6 Temperature sensor status register (DTS_SR)** .................... 1066
- **28.6.7 Temperature sensor interrupt enable register (DTS_ITENR)** ....... 1067
- **28.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR)** .... 1068
- **28.6.9 Temperature sensor option register (DTS_OR)** ................... 1069
- **28.6.10 DTS register map** ..................... 1070

**29 Digital-to-analog converter (DAC)** .......... 1071

**29.1 Introduction** ............................... 1071

**29.2 DAC main features** ....................... 1071

**29.3 DAC implementation** ..................... 1072

**29.4 DAC functional description** ............. 1073
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>29.4.1</td>
<td>DAC block diagram</td>
<td>1073</td>
</tr>
<tr>
<td>29.4.2</td>
<td>DAC pins and internal signals</td>
<td>1074</td>
</tr>
<tr>
<td>29.4.3</td>
<td>DAC channel enable</td>
<td>1076</td>
</tr>
<tr>
<td>29.4.4</td>
<td>DAC data format</td>
<td>1076</td>
</tr>
<tr>
<td>29.4.5</td>
<td>DAC conversion</td>
<td>1077</td>
</tr>
<tr>
<td>29.4.6</td>
<td>DAC output voltage</td>
<td>1077</td>
</tr>
<tr>
<td>29.4.7</td>
<td>DAC trigger selection</td>
<td>1078</td>
</tr>
<tr>
<td>29.4.8</td>
<td>DMA requests</td>
<td>1078</td>
</tr>
<tr>
<td>29.4.9</td>
<td>DAC noise generation</td>
<td>1079</td>
</tr>
<tr>
<td>29.4.10</td>
<td>DAC triangle-wave generation</td>
<td>1080</td>
</tr>
<tr>
<td>29.4.11</td>
<td>DAC channel modes</td>
<td>1081</td>
</tr>
<tr>
<td>29.4.12</td>
<td>DAC channel buffer calibration</td>
<td>1084</td>
</tr>
<tr>
<td>29.4.13</td>
<td>Dual DAC channel conversion modes (if dual channels are available)</td>
<td>1085</td>
</tr>
<tr>
<td>29.5</td>
<td>DAC low-power modes</td>
<td>1089</td>
</tr>
<tr>
<td>29.6</td>
<td>DAC interrupts</td>
<td>1090</td>
</tr>
<tr>
<td>29.7</td>
<td>DAC registers</td>
<td>1091</td>
</tr>
<tr>
<td>29.7.1</td>
<td>DAC control register (DAC_CR)</td>
<td>1091</td>
</tr>
<tr>
<td>29.7.2</td>
<td>DAC software trigger register (DAC_SWTRGR)</td>
<td>1094</td>
</tr>
<tr>
<td>29.7.3</td>
<td>DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)</td>
<td>1095</td>
</tr>
<tr>
<td>29.7.4</td>
<td>DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)</td>
<td>1095</td>
</tr>
<tr>
<td>29.7.5</td>
<td>DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)</td>
<td>1096</td>
</tr>
<tr>
<td>29.7.6</td>
<td>DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)</td>
<td>1096</td>
</tr>
<tr>
<td>29.7.7</td>
<td>DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)</td>
<td>1097</td>
</tr>
<tr>
<td>29.7.8</td>
<td>DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)</td>
<td>1097</td>
</tr>
<tr>
<td>29.7.9</td>
<td>Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)</td>
<td>1098</td>
</tr>
<tr>
<td>29.7.10</td>
<td>Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD)</td>
<td>1098</td>
</tr>
<tr>
<td>29.7.11</td>
<td>Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)</td>
<td>1099</td>
</tr>
<tr>
<td>29.7.12</td>
<td>DAC channel1 data output register (DAC_DOR1)</td>
<td>1099</td>
</tr>
<tr>
<td>29.7.13</td>
<td>DAC channel2 data output register (DAC_DOR2)</td>
<td>1100</td>
</tr>
<tr>
<td>29.7.14</td>
<td>DAC status register (DAC_SR)</td>
<td>1100</td>
</tr>
</tbody>
</table>
30 Voltage reference buffer (VREFBUF) ........................................ 1109

30.1 Introduction ................................................................. 1109
30.2 VREFBUF functional description ...................................... 1109
30.3 VREFBUF registers ........................................................ 1110
   30.3.1 VREFBUF control and status register (VREFBUF_CSR) ...... 1110
   30.3.2 VREFBUF calibration control register (VREFBUF_CCR) .... 1111
   30.3.3 VREFBUF register map ............................................ 1111

31 Comparator (COMP) .......................................................... 1112

31.1 Introduction ............................................................... 1112
31.2 COMP main features ...................................................... 1112
31.3 COMP functional description ......................................... 1113
   31.3.1 COMP block diagram .............................................. 1113
   31.3.2 COMP pins and internal signals ................................ 1114
   31.3.3 COMP reset and clocks .......................................... 1115
   31.3.4 Comparator LOCK mechanism .................................... 1115
   31.3.5 Window comparator ................................................ 1116
   31.3.6 Hysteresis .......................................................... 1116
   31.3.7 Comparator output blanking function ......................... 1116
   31.3.8 Comparator output on GPIOs ..................................... 1117
   31.3.9 Comparator output redirection .................................. 1118
   31.3.10 COMP power and speed modes .................................. 1119
   31.3.11 Scaler function ................................................... 1120
31.4 COMP low-power modes .................................................. 1120
31.5 COMP interrupts .......................................................... 1121
   31.5.1 Interrupt through EXTI block .................................. 1121
   31.5.2 Interrupt through NVIC of the CPU ......................... 1121
## Contents

### 31.6  COMP registers

- 31.6.1 COMP status register (COMP_SR) .................................................. 1122
- 31.6.2 COMP interrupt clear flag register (COMP_ICFR) ............................. 1122
- 31.6.3 COMP option register [alternate] (COMP_OR) ............................... 1123
- 31.6.4 COMP configuration register 1 (COMP_CFRG1) ............................ 1124
- 31.6.5 COMP configuration register 2 (COMP_CFRG2) ............................ 1126
- 31.6.6 COMP register map ................................................................. 1129

### 32  Operational amplifiers (OPAMP) ..................................................... 1130

- 32.1 Introduction .................................................................................. 1130
- 32.2 OPAMP main features ............................................................... 1130
- 32.3 OPAMP functional description .................................................... 1130
  - 32.3.1 OPAMP reset and clocks ......................................................... 1130
  - 32.3.2 Initial configuration ............................................................... 1131
  - 32.3.3 Signal routing ....................................................................... 1131
  - 32.3.4 OPAMP modes ..................................................................... 1132
  - 32.3.5 Calibration ............................................................................ 1139
- 32.4 OPAMP low-power modes ............................................................ 1141
- 32.5 OPAMP PGA gain ....................................................................... 1141
- 32.6 OPAMP registers ....................................................................... 1141
  - 32.6.1 OPAMP1 control/status register (OPAMP1_CSR) .................... 1141
  - 32.6.2 OPAMP1 trimming register in normal mode (OPAMP1_OTR) ... 1143
  - 32.6.3 OPAMP1 trimming register in high-speed mode (OPAMP1_HSOTR) 1144
  - 32.6.4 OPAMP option register (OPAMP_OR) ..................................... 1144
  - 32.6.5 OPAMP2 control/status register (OPAMP2_CSR) ................. 1144
  - 32.6.6 OPAMP2 trimming register in normal mode (OPAMP2_OTR) ... 1146
  - 32.6.7 OPAMP2 trimming register in high-speed mode (OPAMP2_HSOTR) 1147
  - 32.6.8 OPAMP register map ............................................................. 1148

### 33  Digital filter for sigma delta modulators (DFSDM) ............................ 1149

- 33.1 Introduction .................................................................................. 1149
- 33.2 DFSDM main features ............................................................... 1150
- 33.3 DFSDM implementation ............................................................. 1151
- 33.4 DFSDM functional description .................................................... 1152
  - 33.4.1 DFSDM block diagram .......................................................... 1152
  - 33.4.2 DFSDM pins and internal signals ........................................... 1153
### 33.3.3 DFSDM reset and clocks ........................................... 1155
### 33.4.4 Serial channel transceivers .................................... 1156
### 33.4.5 Configuring the input serial interface .................... 1165
### 33.4.6 Parallel data inputs ............................................. 1165
### 33.4.7 Channel selection .............................................. 1168
### 33.4.8 Digital filter configuration ................................... 1168
### 33.4.9 Integrator unit ................................................. 1169
### 33.4.10 Analog watchdog ............................................... 1170
### 33.4.11 Short-circuit detector ......................................... 1172
### 33.4.12 Extreme detector ............................................... 1173
### 33.4.13 Data unit block ................................................ 1173
### 33.4.14 Signed data format ............................................. 1174
### 33.4.15 Launching conversions ........................................ 1175
### 33.4.16 Continuous and fast continuous modes .................... 1175
### 33.4.17 Request precedence ............................................ 1176
### 33.4.18 Power optimization in run mode ............................ 1177

#### 33.5 DFSDM interrupts .................................................. 1177

#### 33.6 DFSDM DMA transfer ............................................... 1179

#### 33.7 DFSDM channel y registers (y=0..7) ............................ 1179
  - 33.7.1 DFSDM channel y configuration register (DFSDM_CHyCFGR1) ... 1179
  - 33.7.2 DFSDM channel y configuration register (DFSDM_CHyCFGR2) ... 1181
  - 33.7.3 DFSDM channel y analog watchdog and short-circuit detector register (DFSDM_CHyAWSCDR) ........................................... 1182
  - 33.7.4 DFSDM channel y watchdog filter data register (DFSDM_CHyWDATR) ................................................................. 1183
  - 33.7.5 DFSDM channel y data input register (DFSDM_CHyDATINR) .... 1183
  - 33.7.6 DFSDM channel y delay register (DFSDM_CHyDLYR) ........ 1184

#### 33.8 DFSDM filter x module registers (x=0..7) .................... 1185
  - 33.8.1 DFSDM filter x control register 1 (DFSDM_FLTxCR1) ........ 1185
  - 33.8.2 DFSDM filter x control register 2 (DFSDM_FLTxCR2) ........ 1188
  - 33.8.3 DFSDM filter x interrupt and status register (DFSDM_FLTxISR) ... 1189
  - 33.8.4 DFSDM filter x interrupt flag clear register (DFSDM_FLTxICR) ... 1191
  - 33.8.5 DFSDM filter x injected channel group selection register (DFSDM_FLTxJCHGR) ............................................................. 1192
  - 33.8.6 DFSDM filter x control register (DFSDM_FLTxFCR) .......... 1192
  - 33.8.7 DFSDM filter x data register for injected group (DFSDM_FLTxJDATAR) ................................................................. 1193
33.8.8 DFSDM filter x data register for the regular channel
(DFSDM_FLTxRDATAR) ........................................... 1194
33.8.9 DFSDM filter x analog watchdog high threshold register
(DFSDM_FLTxAWHTTR) .......................................... 1195
33.8.10 DFSDM filter x analog watchdog low threshold register
(DFSDM_FLTxAWLTR) ........................................... 1195
33.8.11 DFSDM filter x analog watchdog status register
(DFSDM_FLTxAWSR) ............................................. 1196
33.8.12 DFSDM filter x analog watchdog clear flag register
(DFSDM_FLTxAWCFR) ............................................. 1197
33.8.13 DFSDM filter x extremes detector maximum register
(DFSDM_FLTxEXMAX) ............................................. 1197
33.8.14 DFSDM filter x extremes detector minimum register
(DFSDM_FLTxEXMIN) ............................................. 1198
33.8.15 DFSDM filter x conversion timer register (DFSDM_FLTxCNVTIMR). 1198
33.8.16 DFSDM register map ........................................ 1199

34 Digital camera interface (DCMI) .................................. 1214
34.1 Introduction ......................................................... 1214
34.2 DCMI main features ............................................... 1214
34.3 DCMI functional description ...................................... 1214
  34.3.1 DCMI block diagram .......................................... 1215
  34.3.2 DCMI pins and internal signals ................................ 1215
  34.3.3 DCMI clocks .................................................. 1216
  34.3.4 DCMI DMA interface .......................................... 1216
  34.3.5 DCMI physical interface ...................................... 1216
  34.3.6 DCMI synchronization ........................................ 1218
  34.3.7 DCMI capture modes .......................................... 1220
  34.3.8 DCMI crop feature .......................................... 1221
  34.3.9 DCMI JPEG format ........................................... 1222
  34.3.10 DCMI FIFO .................................................. 1222
  34.3.11 DCMI data format description ............................. 1223
34.4 DCMI interrupts .................................................... 1225
34.5 DCMI registers ..................................................... 1226
  34.5.1 DCMI control register (DCMI_CR) ............................ 1226
  34.5.2 DCMI status register (DCMI_SR) .............................. 1228
  34.5.3 DCMI raw interrupt status register (DCMI_RIS) ............ 1229
  34.5.4 DCMI interrupt enable register (DCMI_IER) ................. 1230
  34.5.5 DCMI masked interrupt status register (DCMI_MIS) ....... 1231
34.5.6 DCMI interrupt clear register (DCMI_ICR) ........................................ 1232
34.5.7 DCMI embedded synchronization code register (DCMI_ESCR) .... 1232
34.5.8 DCMI embedded synchronization unmask register (DCMI_ESUR) . 1234
34.5.9 DCMI crop window start (DCMI_CWSTRT) ................................. 1234
34.5.10 DCMI crop window size (DCMI_CWSIZE) ..................................... 1235
34.5.11 DCMI data register (DCMI_DR) .................................................. 1235
34.5.12 DCMI register map .............................................................. 1237

35  Parallel synchronous slave interface (PSSI) ................................. 1238
35.1 Introduction .................................................................................. 1238
35.2 PSSI main features ....................................................................... 1238
35.3 PSSI functional description .......................................................... 1238
  35.3.1 PSSI block diagram .................................................................. 1239
  35.3.2 PSSI pins and internal signals .................................................. 1239
  35.3.3 PSSI clock .............................................................................. 1240
  35.3.4 PSSI data management ........................................................... 1240
  35.3.5 PSSI optional control signals ................................................... 1242
35.4 PSSI interrupts ............................................................................. 1245
35.5 PSSI registers .............................................................................. 1246
  35.5.1 PSSI control register (PSSI_CR) ............................................. 1246
  35.5.2 PSSI status register (PSSI_SR) .............................................. 1247
  35.5.3 PSSI raw interrupt status register (PSSI_RIS) ......................... 1248
  35.5.4 PSSI interrupt enable register (PSSI_IER) .............................. 1249
  35.5.5 PSSI masked interrupt status register (PSSI_MIS) .................. 1249
  35.5.6 PSSI interrupt clear register (PSSI_ICR) ................................ 1250
  35.5.7 PSSI data register (PSSI_DR) ................................................ 1251
  35.5.8 PSSI register map .............................................................. 1252

36  LCD-TFT display controller (LTDC) .............................................. 1253
36.1 Introduction .................................................................................. 1253
36.2 LTDC main features ..................................................................... 1253
36.3 LTDC functional description ......................................................... 1254
  36.3.1 LTDC block diagram ............................................................ 1254
  36.3.2 LCD-TFT internal signals ..................................................... 1254
  36.3.3 LTDC pins and external signal interface ................................. 1255
  36.3.4 LTDC reset and clocks .......................................................... 1255
36.4 LTDC programmable parameters ........................................ 1257
  36.4.1 LTDC global configuration parameters .......................... 1257
  36.4.2 Layer programmable parameters ................................. 1260
36.5 LTDC interrupts .......................................................... 1265
36.6 LTDC programming procedure .......................................... 1265
36.7 LTDC registers ............................................................ 1267
  36.7.1 LTDC synchronization size configuration register (LTDC_SSCR) ... 1267
  36.7.2 LTDC back porch configuration register (LTDC_BPCR) ............. 1267
  36.7.3 LTDC active width configuration register (LTDC_AWCR) .......... 1268
  36.7.4 LTDC total width configuration register (LTDC_TWCR) ......... 1269
  36.7.5 LTDC global control register (LTDC_GCR) ...................... 1269
  36.7.6 LTDC shadow reload configuration register (LTDC_SRCR) ... 1271
  36.7.7 LTDC background color configuration register (LTDC_BCCR) ... 1271
  36.7.8 LTDC interrupt enable register (LTDC_IER) .................. 1272
  36.7.9 LTDC interrupt status register (LTDC_ISR) ................. 1273
  36.7.10 LTDC Interrupt Clear Register (LTDC_ICR) ................ 1273
  36.7.11 LTDC line interrupt position configuration register (LTDC_LIPCR) ... 1274
  36.7.12 LTDC current position status register (LTDC_CPSR) .......... 1274
  36.7.13 LTDC current display status register (LTDC_CDSR) .......... 1275
  36.7.14 LTDC layer x control register (LTDC_LxCR) ............. 1275
  36.7.15 LTDC layer x window horizontal position configuration register (LTDC_LxWHPCR) ... 1276
  36.7.16 LTDC layer x window vertical position configuration register (LTDC_LxWVPCR) ... 1277
  36.7.17 LTDC layer x color keying configuration register (LTDC_LxCKCR) ... 1278
  36.7.18 LTDC layer x pixel format configuration register (LTDC_LxFPCR) ... 1278
  36.7.19 LTDC layer x constant alpha configuration register (LTDC_LxCACR) ... 1279
  36.7.20 LTDC layer x default color configuration register (LTDC_LxDCCR) ... 1280
  36.7.21 LTDC layer x blending factors configuration register (LTDC_LxBFCR) ... 1281
  36.7.22 LTDC layer x color frame buffer address register (LTDC_LxCFBAR) ... 1282
  36.7.23 LTDC layer x color frame buffer length register (LTDC_LxCFBLR) ... 1282
  36.7.24 LTDC layer x color frame buffer line number register (LTDC_LxCFBLNR) ... 1283
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.7.25</td>
<td>LTDC layer x CLUT write register (LTDC_LxCLUTWR)</td>
</tr>
<tr>
<td>36.7.26</td>
<td>LTDC register map</td>
</tr>
<tr>
<td>37</td>
<td>JPEG codec (JPEG)</td>
</tr>
<tr>
<td>37.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>37.2</td>
<td>JPEG codec main features</td>
</tr>
<tr>
<td>37.3</td>
<td>JPEG codec block functional description</td>
</tr>
<tr>
<td>37.3.1</td>
<td>General description</td>
</tr>
<tr>
<td>37.3.2</td>
<td>JPEG internal signals</td>
</tr>
<tr>
<td>37.3.3</td>
<td>JPEG decoding procedure</td>
</tr>
<tr>
<td>37.3.4</td>
<td>JPEG encoding procedure</td>
</tr>
<tr>
<td>37.4</td>
<td>JPEG codec interrupts</td>
</tr>
<tr>
<td>37.5</td>
<td>JPEG codec registers</td>
</tr>
<tr>
<td>37.5.1</td>
<td>JPEG codec control register (JPEG_CONFR0)</td>
</tr>
<tr>
<td>37.5.2</td>
<td>JPEG codec configuration register 1 (JPEG_CONFR1)</td>
</tr>
<tr>
<td>37.5.3</td>
<td>JPEG codec configuration register 2 (JPEG_CONFR2)</td>
</tr>
<tr>
<td>37.5.4</td>
<td>JPEG codec configuration register 3 (JPEG_CONFR3)</td>
</tr>
<tr>
<td>37.5.5</td>
<td>JPEG codec configuration register 4-7 (JPEG_CONFR4-7)</td>
</tr>
<tr>
<td>37.5.6</td>
<td>JPEG control register (JPEG_CR)</td>
</tr>
<tr>
<td>37.5.7</td>
<td>JPEG status register (JPEG_SR)</td>
</tr>
<tr>
<td>37.5.8</td>
<td>JPEG clear flag register (JPEG_CFR)</td>
</tr>
<tr>
<td>37.5.9</td>
<td>JPEG data input register (JPEG_DIR)</td>
</tr>
<tr>
<td>37.5.10</td>
<td>JPEG data output register (JPEG_DOR)</td>
</tr>
<tr>
<td>37.5.11</td>
<td>JPEG quantization memory x (JPEG_QMEMx[y])</td>
</tr>
<tr>
<td>37.5.12</td>
<td>JPEG Huffman min (JPEG_HUFFMIN[4*y+x])</td>
</tr>
<tr>
<td>37.5.13</td>
<td>JPEG Huffman base (JPEG_HUFFBASE[y])</td>
</tr>
<tr>
<td>37.5.14</td>
<td>JPEG Huffman symbol (JPEG_HUFFSYMBl[y])</td>
</tr>
<tr>
<td>37.5.15</td>
<td>JPEG DHT memory (JPEG_DHTMEMl[y])</td>
</tr>
<tr>
<td>37.5.16</td>
<td>JPEG Huffman encoder ACx (JPEG_HUFFENC_ACx[y])</td>
</tr>
<tr>
<td>37.5.17</td>
<td>JPEG Huffman encoder DCx (JPEG_HUFFENC_DCx[y])</td>
</tr>
<tr>
<td>37.5.18</td>
<td>JPEG codec register map</td>
</tr>
<tr>
<td>38</td>
<td>True random number generator (RNG)</td>
</tr>
<tr>
<td>38.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>38.2</td>
<td>RNG main features</td>
</tr>
<tr>
<td>38.3</td>
<td>RNG functional description</td>
</tr>
</tbody>
</table>
### 38.3 RNG

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>38.3.1 RNG block diagram</td>
<td>1309</td>
</tr>
<tr>
<td>38.3.2 RNG internal signals</td>
<td>1309</td>
</tr>
<tr>
<td>38.3.3 Random number generation</td>
<td>1310</td>
</tr>
<tr>
<td>38.3.4 RNG initialization</td>
<td>1312</td>
</tr>
<tr>
<td>38.3.5 RNG operation</td>
<td>1314</td>
</tr>
<tr>
<td>38.3.6 RNG clocking</td>
<td>1315</td>
</tr>
<tr>
<td>38.3.7 Error management</td>
<td>1315</td>
</tr>
<tr>
<td>38.3.8 RNG low-power usage</td>
<td>1316</td>
</tr>
<tr>
<td>38.4 RNG interrupts</td>
<td>1316</td>
</tr>
<tr>
<td>38.5 RNG processing time</td>
<td>1317</td>
</tr>
<tr>
<td>38.6 RNG entropy source validation</td>
<td>1318</td>
</tr>
<tr>
<td>38.6.1 Introduction</td>
<td>1318</td>
</tr>
<tr>
<td>38.6.2 Validation conditions</td>
<td>1318</td>
</tr>
<tr>
<td>38.6.3 Data collection</td>
<td>1318</td>
</tr>
<tr>
<td>38.7 RNG registers</td>
<td>1319</td>
</tr>
<tr>
<td>38.7.1 RNG control register (RNG_CR)</td>
<td>1319</td>
</tr>
<tr>
<td>38.7.2 RNG status register (RNG_SR)</td>
<td>1321</td>
</tr>
<tr>
<td>38.7.3 RNG data register (RNG_DR)</td>
<td>1322</td>
</tr>
<tr>
<td>38.7.4 RNG health test control register (RNG_HTCR)</td>
<td>1322</td>
</tr>
<tr>
<td>38.7.5 RNG register map</td>
<td>1323</td>
</tr>
</tbody>
</table>

### 39 Cryptographic processor (CRYP)

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>39.1 Introduction</td>
<td>1324</td>
</tr>
<tr>
<td>39.2 CRYP main features</td>
<td>1324</td>
</tr>
<tr>
<td>39.3 CRYP functional description</td>
<td>1326</td>
</tr>
<tr>
<td>39.3.1 CRYP block diagram</td>
<td>1326</td>
</tr>
<tr>
<td>39.3.2 CRYP internal signals</td>
<td>1327</td>
</tr>
<tr>
<td>39.3.3 CRYP DES/TDES cryptographic core</td>
<td>1327</td>
</tr>
<tr>
<td>39.3.4 CRYP AES cryptographic core</td>
<td>1328</td>
</tr>
<tr>
<td>39.3.5 CRYP procedure to perform a cipher operation</td>
<td>1334</td>
</tr>
<tr>
<td>39.3.6 CRYP busy state</td>
<td>1336</td>
</tr>
<tr>
<td>39.3.7 Preparing the CRYP AES key for decryption</td>
<td>1336</td>
</tr>
<tr>
<td>39.3.8 CRYP stealing and data padding</td>
<td>1337</td>
</tr>
<tr>
<td>39.3.9 CRYP suspend/resume operations</td>
<td>1339</td>
</tr>
<tr>
<td>39.3.10 CRYP DES/TDES basic chaining modes (ECB, CBC)</td>
<td>1340</td>
</tr>
<tr>
<td>39.3.11 CRYP AES basic chaining modes (ECB, CBC)</td>
<td>1345</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>39.3.12</td>
<td>CRYP AES counter mode (AES-CTR)</td>
</tr>
<tr>
<td>39.3.13</td>
<td>CRYP AES Galois/counter mode (GCM)</td>
</tr>
<tr>
<td>39.3.14</td>
<td>CRYP AES Galois message authentication code (GMAC)</td>
</tr>
<tr>
<td>39.3.15</td>
<td>CRYP AES Counter with CBC-MAC (CCM)</td>
</tr>
<tr>
<td>39.3.16</td>
<td>CRYP data registers and data swapping</td>
</tr>
<tr>
<td>39.3.17</td>
<td>CRYP key registers</td>
</tr>
<tr>
<td>39.3.18</td>
<td>CRYP initialization vector registers</td>
</tr>
<tr>
<td>39.3.19</td>
<td>CRYP DMA interface</td>
</tr>
<tr>
<td>39.3.20</td>
<td>CRYP error management</td>
</tr>
<tr>
<td>39.4</td>
<td>CRYP interrupts</td>
</tr>
<tr>
<td>39.5</td>
<td>CRYP processing time</td>
</tr>
<tr>
<td>39.6</td>
<td>CRYP registers</td>
</tr>
<tr>
<td>39.6.1</td>
<td>CRYP control register (CRYP_CR)</td>
</tr>
<tr>
<td>39.6.2</td>
<td>CRYP status register (CRYP_SR)</td>
</tr>
<tr>
<td>39.6.3</td>
<td>CRYP data input register (CRYP_DIN)</td>
</tr>
<tr>
<td>39.6.4</td>
<td>CRYP data output register (CRYP_DOUT)</td>
</tr>
<tr>
<td>39.6.5</td>
<td>CRYP DMA control register (CRYP_DMACR)</td>
</tr>
<tr>
<td>39.6.6</td>
<td>CRYP interrupt mask set/clear register (CRYP_IMSCR)</td>
</tr>
<tr>
<td>39.6.7</td>
<td>CRYP raw interrupt status register (CRYP_RISR)</td>
</tr>
<tr>
<td>39.6.8</td>
<td>CRYP masked interrupt status register (CRYP_MISR)</td>
</tr>
<tr>
<td>39.6.9</td>
<td>CRYP key register 0L (CRYP_K0LR)</td>
</tr>
<tr>
<td>39.6.10</td>
<td>CRYP key register 0R (CRYP_K0RR)</td>
</tr>
<tr>
<td>39.6.11</td>
<td>CRYP key register 1L (CRYP_K1LR)</td>
</tr>
<tr>
<td>39.6.12</td>
<td>CRYP key register 1R (CRYP_K1RR)</td>
</tr>
<tr>
<td>39.6.13</td>
<td>CRYP key register 2L (CRYP_K2LR)</td>
</tr>
<tr>
<td>39.6.14</td>
<td>CRYP key register 2R (CRYP_K2RR)</td>
</tr>
<tr>
<td>39.6.15</td>
<td>CRYP key register 3L (CRYP_K3LR)</td>
</tr>
<tr>
<td>39.6.16</td>
<td>CRYP key register 3R (CRYP_K3RR)</td>
</tr>
<tr>
<td>39.6.17</td>
<td>CRYP initialization vector register 0L (CRYP_IV0LR)</td>
</tr>
<tr>
<td>39.6.18</td>
<td>CRYP initialization vector register 0R (CRYP_IV0RR)</td>
</tr>
<tr>
<td>39.6.19</td>
<td>CRYP initialization vector register 1L (CRYP_IV1LR)</td>
</tr>
<tr>
<td>39.6.20</td>
<td>CRYP initialization vector register 1R (CRYP_IV1RR)</td>
</tr>
<tr>
<td>39.6.21</td>
<td>CRYP context swap GCM-CCM registers (CRYP_CSGMCCMxR)</td>
</tr>
<tr>
<td>39.6.22</td>
<td>CRYP context swap GCM registers (CRYP_CSGCMxR)</td>
</tr>
<tr>
<td>39.6.23</td>
<td>CRYP register map</td>
</tr>
</tbody>
</table>

### Hash processor (HASH)

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>Hash processor (HASH)</td>
</tr>
</tbody>
</table>

1350  1354  1359  1360  1366  1370  1371  1372  1374  1376  1377  1379  1380  1381  1382  1383  1384  1385  1386  1387  1388  1389  1390  1393
40.1 Introduction .................................................. 1393
40.2 HASH main features ........................................ 1393
40.3 HASH implementation ....................................... 1394
40.4 HASH functional description .............................. 1394
  40.4.1 HASH block diagram .................................. 1394
  40.4.2 HASH internal signals ................................. 1395
  40.4.3 About secure hash algorithms ....................... 1395
  40.4.4 Message data feeding ................................ 1395
  40.4.5 Message digest computing ......................... 1397
  40.4.6 Message padding .................................... 1398
  40.4.7 HMAC operation ..................................... 1400
  40.4.8 HASH suspend/resume operations ................. 1402
  40.4.9 HASH DMA interface .................................. 1404
  40.4.10 HASH error management ......................... 1404
40.5 HASH interrupts ........................................... 1404
40.6 HASH processing time ...................................... 1405
40.7 HASH registers ............................................ 1406
  40.7.1 HASH control register (HASH_CR) ................. 1406
  40.7.2 HASH data input register (HASH_DIN) ............. 1409
  40.7.3 HASH start register (HASH_STR) .................. 1410
  40.7.4 HASH digest registers ............................... 1411
  40.7.5 HASH interrupt enable register (HASH_IMR) ....... 1412
  40.7.6 HASH status register (HASH_SR) .................. 1413
  40.7.7 HASH context swap registers ....................... 1413
  40.7.8 HASH register map .................................. 1415

41 On-The-Fly Decryption engine - AXI (OTFDEC) ............ 1416
41.1 Introduction ................................................ 1416
41.2 OTFDEC main features .................................... 1416
41.3 OTFDEC functional description ........................... 1417
  41.3.1 OTFDEC block diagram ................................ 1417
  41.3.2 OTFDEC internal signals ............................ 1418
  41.3.3 OTFDEC on-the-fly decryption ..................... 1418
  41.3.4 AES in counter mode decryption ................. 1419
  41.3.5 Flow control management ......................... 1421
  41.3.6 OTFDEC error management ......................... 1423
41.4 OTFDEC interrupts ............................................. 1423
41.5 OTFDEC application information ............................... 1424
  41.5.1 OTFDEC initialization process ............................ 1424
  41.5.2 OTFDEC and power management ............................ 1425
  41.5.3 Encrypting for OTFDEC ................................. 1425
  41.5.4 OTFDEC Key CRC source code .......................... 1426
41.6 OTFDEC registers ............................................. 1427
  41.6.1 OTFDEC region x configuration register (OTFDEC_RxCFGR) ............................................. 1427
  41.6.2 OTFDEC region x start address register (OTFDEC2_RxSTARTADDR) ............................................. 1428
  41.6.3 OTFDEC region x end address register (OTFDEC_RxENDADDR) ............................................. 1428
  41.6.4 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0) ............................................. 1429
  41.6.5 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1) ............................................. 1429
  41.6.6 OTFDEC region x key register 0 (OTFDEC_RxKEYR0) ............................................. 1430
  41.6.7 OTFDEC region x key register 1 (OTFDEC_RxKEYR1) ............................................. 1430
  41.6.8 OTFDEC region x key register 2 (OTFDEC_RxKEYR2) ............................................. 1430
  41.6.9 OTFDEC region x key register 3 (OTFDEC_RxKEYR3) ............................................. 1431
  41.6.10 OTFDEC interrupt status register (OTFDEC_ISR) ............................................. 1431
  41.6.11 OTFDEC interrupt clear register (OTFDEC_ICR) ............................................. 1433
  41.6.12 OTFDEC interrupt enable register (OTFDEC_IER) ............................................. 1433
  41.6.13 OTFDEC register map .................................... 1435

42 Advanced-control timers (TIM1/TIM8) .......................... 1439
  42.1 TIM1/TIM8 introduction ..................................... 1439
  42.2 TIM1/TIM8 main features .................................... 1439
  42.3 TIM1/TIM8 functional description ............................ 1441
    42.3.1 Time-base unit ........................................ 1441
    42.3.2 Counter modes ........................................ 1443
    42.3.3 Repetition counter .................................... 1454
    42.3.4 External trigger input ................................. 1456
    42.3.5 Clock selection ....................................... 1457
    42.3.6 Capture/compare channels ............................. 1461
    42.3.7 Input capture mode .................................... 1463
    42.3.8 PWM input mode ..................................... 1464
    42.3.9 Forced output mode .................................... 1465
    42.3.10 Output compare mode ................................ 1466
    42.3.11 PWM mode ........................................... 1467
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>42.3.12</td>
<td>Asymmetric PWM mode</td>
<td>1470</td>
</tr>
<tr>
<td>42.3.13</td>
<td>Combined PWM mode</td>
<td>1471</td>
</tr>
<tr>
<td>42.3.14</td>
<td>Combined 3-phase PWM mode</td>
<td>1472</td>
</tr>
<tr>
<td>42.3.15</td>
<td>Complementary outputs and dead-time insertion</td>
<td>1473</td>
</tr>
<tr>
<td>42.3.16</td>
<td>Using the break function</td>
<td>1475</td>
</tr>
<tr>
<td>42.3.17</td>
<td>Bidirectional break inputs</td>
<td>1481</td>
</tr>
<tr>
<td>42.3.18</td>
<td>Clearing the OCxREF signal on an external event</td>
<td>1482</td>
</tr>
<tr>
<td>42.3.19</td>
<td>6-step PWM generation</td>
<td>1484</td>
</tr>
<tr>
<td>42.3.20</td>
<td>One-pulse mode</td>
<td>1485</td>
</tr>
<tr>
<td>42.3.21</td>
<td>Retriggerable one pulse mode (OPM)</td>
<td>1486</td>
</tr>
<tr>
<td>42.3.22</td>
<td>Encoder interface mode</td>
<td>1487</td>
</tr>
<tr>
<td>42.3.23</td>
<td>UIF bit remapping</td>
<td>1489</td>
</tr>
<tr>
<td>42.3.24</td>
<td>Timer input XOR function</td>
<td>1490</td>
</tr>
<tr>
<td>42.3.25</td>
<td>Interfacing with Hall sensors</td>
<td>1490</td>
</tr>
<tr>
<td>42.3.26</td>
<td>Timer synchronization</td>
<td>1493</td>
</tr>
<tr>
<td>42.3.27</td>
<td>ADC synchronization</td>
<td>1497</td>
</tr>
<tr>
<td>42.3.28</td>
<td>DMA burst mode</td>
<td>1497</td>
</tr>
<tr>
<td>42.3.29</td>
<td>Debug mode</td>
<td>1498</td>
</tr>
<tr>
<td>42.4</td>
<td>TIM1/TIM8 registers</td>
<td>1499</td>
</tr>
<tr>
<td>42.4.1</td>
<td>TIMx control register 1 (TIMx_CR1)(x = 1, 8)</td>
<td>1499</td>
</tr>
<tr>
<td>42.4.2</td>
<td>TIMx control register 2 (TIMx_CR2)(x = 1, 8)</td>
<td>1500</td>
</tr>
<tr>
<td>42.4.3</td>
<td>TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)</td>
<td>1503</td>
</tr>
<tr>
<td>42.4.4</td>
<td>TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8)</td>
<td>1505</td>
</tr>
<tr>
<td>42.4.5</td>
<td>TIMx status register (TIMx_SR)(x = 1, 8)</td>
<td>1507</td>
</tr>
<tr>
<td>42.4.6</td>
<td>TIMx event generation register (TIMx_EGR)(x = 1, 8)</td>
<td>1509</td>
</tr>
<tr>
<td>42.4.7</td>
<td>TIMx capture/compare mode register 1 [alternate]</td>
<td>1510</td>
</tr>
<tr>
<td></td>
<td>(TIMx_CCMR1)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.8</td>
<td>TIMx capture/compare mode register 1 [alternate]</td>
<td>1511</td>
</tr>
<tr>
<td></td>
<td>(TIMx_CCMR1)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.9</td>
<td>TIMx capture/compare mode register 2 [alternate]</td>
<td>1514</td>
</tr>
<tr>
<td></td>
<td>(TIMx_CCMR2)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.10</td>
<td>TIMx capture/compare mode register 2 [alternate]</td>
<td>1515</td>
</tr>
<tr>
<td></td>
<td>(TIMx_CCMR2)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.11</td>
<td>TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)</td>
<td>1517</td>
</tr>
<tr>
<td>42.4.12</td>
<td>TIMx counter (TIMx_CNT)(x = 1, 8)</td>
<td>1520</td>
</tr>
<tr>
<td>42.4.13</td>
<td>TIMx prescaler (TIMx_PSC)(x = 1, 8)</td>
<td>1520</td>
</tr>
<tr>
<td>42.4.14</td>
<td>TIMx auto-reload register (TIMx_ARR)(x = 1, 8)</td>
<td>1520</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>42.4.15</td>
<td>TIMx repetition counter register (TIMx_RCR)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.16</td>
<td>TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.17</td>
<td>TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.18</td>
<td>TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.19</td>
<td>TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.20</td>
<td>TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.21</td>
<td>TIMx DMA control register (TIMx_DCR)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.22</td>
<td>TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.23</td>
<td>TIMx capture/compare mode register 3 (TIMx_CCMR3)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.24</td>
<td>TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.25</td>
<td>TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8)</td>
<td></td>
</tr>
<tr>
<td>42.4.26</td>
<td>TIM1 alternate function option register 1 (TIM1_AF1)</td>
<td></td>
</tr>
<tr>
<td>42.4.27</td>
<td>TIM1 Alternate function register 2 (TIM1_AF2)</td>
<td></td>
</tr>
<tr>
<td>42.4.28</td>
<td>TIM8 Alternate function option register 1 (TIM8_AF1)</td>
<td></td>
</tr>
<tr>
<td>42.4.29</td>
<td>TIM8 Alternate function option register 2 (TIM8_AF2)</td>
<td></td>
</tr>
<tr>
<td>42.4.30</td>
<td>TIM1 timer input selection register (TIM1_TISEL)</td>
<td></td>
</tr>
<tr>
<td>42.4.31</td>
<td>TIM8 timer input selection register (TIM8_TISEL)</td>
<td></td>
</tr>
<tr>
<td>42.4.32</td>
<td>TIM1 register map</td>
<td></td>
</tr>
<tr>
<td>42.4.33</td>
<td>TIM8 register map</td>
<td></td>
</tr>
</tbody>
</table>

### 43 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>43.1</td>
<td>TIM2/TIM3/TIM4/TIM5 introduction</td>
</tr>
<tr>
<td>43.2</td>
<td>TIM2/TIM3/TIM4/TIM5 main features</td>
</tr>
<tr>
<td>43.3</td>
<td>TIM2/TIM3/TIM4/TIM5 functional description</td>
</tr>
<tr>
<td>43.3.1</td>
<td>Time-base unit</td>
</tr>
<tr>
<td>43.3.2</td>
<td>Counter modes</td>
</tr>
<tr>
<td>43.3.3</td>
<td>Clock selection</td>
</tr>
<tr>
<td>43.3.4</td>
<td>Capture/Compare channels</td>
</tr>
<tr>
<td>43.3.5</td>
<td>Input capture mode</td>
</tr>
<tr>
<td>43.3.6</td>
<td>PWM input mode</td>
</tr>
<tr>
<td>43.3.7</td>
<td>Forced output mode</td>
</tr>
<tr>
<td>43.3.8</td>
<td>Output compare mode</td>
</tr>
<tr>
<td>43.3.9</td>
<td>PWM mode</td>
</tr>
<tr>
<td>43.3.10</td>
<td>Asymmetric PWM mode</td>
</tr>
<tr>
<td>43.3.11</td>
<td>Combined PWM mode</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>43.3.12</td>
<td>Clearing the OCxREF signal on an external event</td>
</tr>
<tr>
<td>43.3.13</td>
<td>One-pulse mode</td>
</tr>
<tr>
<td>43.3.14</td>
<td>Retriggerable one pulse mode (OPM)</td>
</tr>
<tr>
<td>43.3.15</td>
<td>Encoder interface mode</td>
</tr>
<tr>
<td>43.3.16</td>
<td>UIF bit remapping</td>
</tr>
<tr>
<td>43.3.17</td>
<td>Timer input XOR function</td>
</tr>
<tr>
<td>43.3.18</td>
<td>Timers and external trigger synchronization</td>
</tr>
<tr>
<td>43.3.19</td>
<td>Timer synchronization</td>
</tr>
<tr>
<td>43.3.20</td>
<td>DMA burst mode</td>
</tr>
<tr>
<td>43.3.21</td>
<td>Debug mode</td>
</tr>
<tr>
<td>43.4</td>
<td>TIM2/TIM3/TIM4/TIM5 registers</td>
</tr>
<tr>
<td>43.4.1</td>
<td>TIMx control register 1 (TIMx_CR1)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.2</td>
<td>TIMx control register 2 (TIMx_CR2)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.3</td>
<td>TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.4</td>
<td>TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.5</td>
<td>TIMx status register (TIMx_SR)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.6</td>
<td>TIMx event generation register (TIMx_EGR)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.7</td>
<td>TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.8</td>
<td>TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.9</td>
<td>TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2) (x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.10</td>
<td>TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2) (x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.11</td>
<td>TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.12</td>
<td>TIMx counter [alternate] (TIMx_CNT)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.13</td>
<td>TIMx counter [alternate] (TIMx_CNT)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.14</td>
<td>TIMx prescaler (TIMx_PSC)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.15</td>
<td>TIMx auto-reload register (TIMx_ARR)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.16</td>
<td>TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.17</td>
<td>TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.18</td>
<td>TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.19</td>
<td>TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.20</td>
<td>TIMx DMA control register (TIMx_DCR)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.21</td>
<td>TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)</td>
</tr>
<tr>
<td>43.4.22</td>
<td>TIM2 alternate function option register 1 (TIM2_AF1)</td>
</tr>
</tbody>
</table>
43.4.23 TIM3 alternate function option register 1 (TIM3_AF1) ........................................ 1612
43.4.24 TIM4 alternate function option register 1 (TIM4_AF1) ........................................ 1612
43.4.25 TIM5 alternate function option register 1 (TIM5_AF1) ........................................ 1613
43.4.26 TIM2 timer input selection register (TIM2_TISEL) ........................................... 1613
43.4.27 TIM3 timer input selection register (TIM3_TISEL) ........................................... 1614
43.4.28 TIM4 timer input selection register (TIM4_TISEL) ........................................... 1615
43.4.29 TIM5 timer input selection register (TIM5_TISEL) ........................................... 1616
43.4.30 TIMx register map ............................................................. 1617

44  General-purpose timers (TIM12/TIM13/TIM14) ............................................ 1620

44.1 TIM12/TIM13/TIM14 introduction ................................................................. 1620
44.2 TIM12/TIM13/TIM14 main features ............................................................... 1620
  44.2.1 TIM12 main features ................................................................. 1620
  44.2.2 TIM13/TIM14 main features .............................................................. 1621
44.3 TIM12/TIM13/TIM14 functional description .................................................. 1623
  44.3.1 Time-base unit ................................................................. 1623
  44.3.2 Counter modes ................................................................. 1625
  44.3.3 Clock selection ................................................................. 1628
  44.3.4 Capture/compare channels .......................................................... 1630
  44.3.5 Input capture mode ............................................................... 1632
  44.3.6 PWM input mode (only for TIM12) .................................................... 1633
  44.3.7 Forced output mode ................................................................. 1634
  44.3.8 Output compare mode ................................................................. 1635
  44.3.9 PWM mode ................................................................. 1636
  44.3.10 Combined PWM mode (TIM12 only) .................................................... 1637
  44.3.11 One-pulse mode ................................................................. 1638
  44.3.12 Retriggerable one pulse mode (OPM) (TIM12 only) ...................................... 1640
  44.3.13 UIF bit remapping ................................................................. 1641
  44.3.14 Timer input XOR function ........................................................ 1641
  44.3.15 TIM12 external trigger synchronization ................................................. 1641
  44.3.16 Slave mode – combined reset + trigger mode ........................................... 1644
  44.3.17 Timer synchronization (TIM12) ....................................................... 1645
  44.3.18 Debug mode ................................................................. 1645

44.4 TIM12 registers ................................................................. 1645
  44.4.1 TIM12 control register 1 (TIM12_CR1) .................................................... 1645
  44.4.2 TIM12 control register 2 (TIM12_CR2) .................................................... 1646
  44.4.3 TIM12 slave mode control register (TIM12_SMCR) ...................................... 1647
44.4.4 TIM12 Interrupt enable register (TIM12_DIER) .................................. 1649
44.4.5 TIM12 status register (TIM12_SR) ..................................................... 1649
44.4.6 TIM12 event generation register (TIM12_EGR) ................................. 1650
44.4.7 TIM12 capture/compare mode register 1 [alternate] (TIM12_CCMR1) ................................. 1651
44.4.8 TIM12 capture/compare mode register 1 [alternate] (TIM12_CCMR1) ................................. 1652
44.4.9 TIM12 capture/compare enable register (TIM12_CCER) ....................... 1655
44.4.10 TIM12 counter (TIM12_CNT) ......................................................... 1656
44.4.11 TIM12 prescaler (TIM12_PSC) ....................................................... 1657
44.4.12 TIM12 auto-reload register (TIM12_ARR) ....................................... 1657
44.4.13 TIM12 capture/compare register 1 (TIM12_CCR1) ............................. 1657
44.4.14 TIM12 capture/compare register 2 (TIM12_CCR2) ........................... 1658
44.4.15 TIM12 timer input selection register (TIM12_TISEL) ......................... 1658
44.4.16 TIM12 register map ................................................................. 1659

44.5 TIM13/TIM14 registers ............................................................. 1661
44.5.1 TIMx control register 1 (TIMx_CR1)(x = 13 to 14) .......................... 1661
44.5.2 TIMx Interrupt enable register (TIMx_DIER)(x = 13 to 14) ............... 1662
44.5.3 TIMx status register (TIMx_SR)(x = 13 to 14) ................................ 1662
44.5.4 TIMx event generation register (TIMx_EGR)(x = 13 to 14) ............... 1663
44.5.5 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 13 to 14) ............... 1664
44.5.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 13 to 14) ............... 1665
44.5.7 TIMx capture/compare enable register (TIMx_CCER)(x = 13 to 14) ............... 1667
44.5.8 TIMx counter (TIMx_CNT)(x = 13 to 14) ...................................... 1668
44.5.9 TIMx prescaler (TIMx_PSC)(x = 13 to 14) .................................... 1668
44.5.10 TIMx auto-reload register (TIMx_ARR)(x = 13 to 14) ................. 1668
44.5.11 TIMx capture/compare register 1 (TIMx_CCR1)(x = 13 to 14) .......... 1669
44.5.12 TIM13 timer input selection register (TIM13_TISEL) ..................... 1669
44.5.13 TIM14 timer input selection register (TIM14_TISEL) ..................... 1669
44.5.14 TIM13/TIM14 register map ..................................................... 1670

45 General-purpose timers (TIM15/TIM16/TIM17) .................................. 1672
45.1 TIM15/TIM16/TIM17 introduction .................................................. 1672
45.2 TIM15 main features ............................................................... 1672
45.3 TIM16/TIM17 main features ....................................................... 1673
45.4 TIM15/TIM16/TIM17 functional description ............................................. 1676
  45.4.1 Time-base unit ................................................................. 1676
  45.4.2 Counter modes .................................................................. 1678
  45.4.3 Repetition counter .............................................................. 1682
  45.4.4 Clock selection .................................................................. 1683
  45.4.5 Capture/compare channels .................................................... 1685
  45.4.6 Input capture mode .............................................................. 1687
  45.4.7 PWM input mode (only for TIM15) .......................................... 1688
  45.4.8 Forced output mode ............................................................ 1689
  45.4.9 Output compare mode ........................................................... 1690
  45.4.10 PWM mode .................................................................... 1691
  45.4.11 Combined PWM mode (TIM15 only) ...................................... 1692
  45.4.12 Complementary outputs and dead-time insertion ..................... 1693
  45.4.13 Using the break function ...................................................... 1695
  45.4.14 Bidirectional break inputs .................................................... 1700
  45.4.15 One-pulse mode ................................................................. 1702
  45.4.16 Retriggerable one pulse mode (OPM) (TIM15 only) .................. 1704
  45.4.17 UIF bit remapping ............................................................... 1704
  45.4.18 Timer input XOR function (TIM15 only) ................................. 1706
  45.4.19 External trigger synchronization (TIM15 only) ....................... 1707
  45.4.20 Slave mode – combined reset + trigger mode ........................... 1709
  45.4.21 DMA burst mode ............................................................... 1709
  45.4.22 Timer synchronization (TIM15) ............................................. 1711
  45.4.23 Debug mode ...................................................................... 1711

45.5 TIM15 registers ........................................................................ 1712
  45.5.1 TIM15 control register 1 (TIM15_CR1) ....................................... 1712
  45.5.2 TIM15 control register 2 (TIM15_CR2) ..................................... 1713
  45.5.3 TIM15 slave mode control register (TIM15_SMCR) .................. 1715
  45.5.4 TIM15 DMA/interrupt enable register (TIM15_DIER) ............... 1716
  45.5.5 TIM15 status register (TIM15_SR) .......................................... 1717
  45.5.6 TIM15 event generation register (TIM15_EGR) ......................... 1719
  45.5.7 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) ................................................................. 1720
  45.5.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) ................................................................. 1721
  45.5.9 TIM15 capture/compare enable register (TIM15_CCER) ............ 1724
  45.5.10 TIM15 counter (TIM15_CNT) ............................................... 1727
45.5.11 TIM15 prescaler (TIM15_PSC) .............................................. 1727
45.5.12 TIM15 auto-reload register (TIM15_ARR) .............................. 1727
45.5.13 TIM15 repetition counter register (TIM15_RCR) ....................... 1728
45.5.14 TIM15 capture/compare register 1 (TIM15_CCR1) .................... 1728
45.5.15 TIM15 capture/compare register 2 (TIM15_CCR2) .................... 1729
45.5.16 TIM15 break and dead-time register (TIM15_BDTR) .................. 1729
45.5.17 TIM15 DMA control register (TIM15_DCR) ............................ 1732
45.5.18 TIM15 DMA address for full transfer (TIM15_DMAR) ................. 1732
45.5.19 TIM15 alternate register 1 (TIM15_AF1) ................................ 1733
45.5.20 TIM15 input selection register (TIM15_TISEL) ....................... 1734
45.5.21 TIM15 register map ....................................................... 1735

45.6 TIM16/TIM17 registers ......................................................... 1738
45.6.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17) .................... 1738
45.6.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17) .................... 1739
45.6.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) 1740
45.6.4 TIMx status register (TIMx_SR)(x = 16 to 17) ....................... 1741
45.6.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17) .......... 1742
45.6.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) 1743
45.6.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) 1744
45.6.8 TIMx capture/compare enable register (TIMx_CCRER)(x = 16 to 17) 1746
45.6.9 TIMx counter (TIMx_CNT)(x = 16 to 17) ............................... 1748
45.6.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17) .... ........................ 1749
45.6.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17) ............... 1749
45.6.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) ....... 1750
45.6.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) ...... 1750
45.6.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) .. 1751
45.6.15 TIMx DMA control register (TIMx_DCR)(x = 16 to 17) ............. 1754
45.6.16 TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17) .. 1754
45.6.17 TIM16 alternate function register 1 (TIM16_AF1) ..................... 1755
45.6.18 TIM16 input selection register (TIM16_TISEL) ....................... 1756
45.6.19 TIM17 alternate function register 1 (TIM17_AF1) ..................... 1757
45.6.20 TIM17 input selection register (TIM17_TISEL) ....................... 1758
45.6.21 TIM16/TIM17 register map ............................................ 1759

46 Basic timers (TIM6/TIM7) ..................................................... 1761
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>46.1</td>
<td>TIM6/TIM7 introduction</td>
<td>1761</td>
</tr>
<tr>
<td>46.2</td>
<td>TIM6/TIM7 main features</td>
<td>1761</td>
</tr>
<tr>
<td>46.3</td>
<td>TIM6/TIM7 functional description</td>
<td>1761</td>
</tr>
<tr>
<td>46.3.1</td>
<td>Time-base unit</td>
<td>1762</td>
</tr>
<tr>
<td>46.3.2</td>
<td>Counting mode</td>
<td>1764</td>
</tr>
<tr>
<td>46.3.3</td>
<td>UIF bit remapping</td>
<td>1767</td>
</tr>
<tr>
<td>46.3.4</td>
<td>Clock source</td>
<td>1767</td>
</tr>
<tr>
<td>46.3.5</td>
<td>Debug mode</td>
<td>1768</td>
</tr>
<tr>
<td>46.4</td>
<td>TIM6/TIM7 registers</td>
<td>1768</td>
</tr>
<tr>
<td>46.4.1</td>
<td>TIMx control register 1 (TIMx_CR1)(x = 6 to 7)</td>
<td>1768</td>
</tr>
<tr>
<td>46.4.2</td>
<td>TIMx control register 2 (TIMx_CR2)(x = 6 to 7)</td>
<td>1770</td>
</tr>
<tr>
<td>46.4.3</td>
<td>TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)</td>
<td>1770</td>
</tr>
<tr>
<td>46.4.4</td>
<td>TIMx status register (TIMx_SR)(x = 6 to 7)</td>
<td>1771</td>
</tr>
<tr>
<td>46.4.5</td>
<td>TIMx event generation register (TIMx_EGR)(x = 6 to 7)</td>
<td>1771</td>
</tr>
<tr>
<td>46.4.6</td>
<td>TIMx counter (TIMx_CNT)(x = 6 to 7)</td>
<td>1771</td>
</tr>
<tr>
<td>46.4.7</td>
<td>TIMx prescaler (TIMx_PSC)(x = 6 to 7)</td>
<td>1772</td>
</tr>
<tr>
<td>46.4.8</td>
<td>TIMx auto-reload register (TIMx_ARR)(x = 6 to 7)</td>
<td>1772</td>
</tr>
<tr>
<td>46.4.9</td>
<td>TIMx register map</td>
<td>1773</td>
</tr>
</tbody>
</table>

**47 Low-power timer (LPTIM)**

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>47.1</td>
<td>Introduction</td>
<td>1774</td>
</tr>
<tr>
<td>47.2</td>
<td>LPTIM main features</td>
<td>1774</td>
</tr>
<tr>
<td>47.3</td>
<td>LPTIM implementation</td>
<td>1775</td>
</tr>
<tr>
<td>47.4</td>
<td>LPTIM functional description</td>
<td>1775</td>
</tr>
<tr>
<td>47.4.1</td>
<td>LPTIM block diagram</td>
<td>1775</td>
</tr>
<tr>
<td>47.4.2</td>
<td>LPTIM pins and internal signals</td>
<td>1776</td>
</tr>
<tr>
<td>47.4.3</td>
<td>LPTIM input and trigger mapping</td>
<td>1777</td>
</tr>
<tr>
<td>47.4.4</td>
<td>LPTIM reset and clocks</td>
<td>1779</td>
</tr>
<tr>
<td>47.4.5</td>
<td>Glitch filter</td>
<td>1779</td>
</tr>
<tr>
<td>47.4.6</td>
<td>Prescaler</td>
<td>1780</td>
</tr>
<tr>
<td>47.4.7</td>
<td>Trigger multiplexer</td>
<td>1780</td>
</tr>
<tr>
<td>47.4.8</td>
<td>Operating mode</td>
<td>1781</td>
</tr>
<tr>
<td>47.4.9</td>
<td>Timeout function</td>
<td>1783</td>
</tr>
<tr>
<td>47.4.10</td>
<td>Waveform generation</td>
<td>1783</td>
</tr>
<tr>
<td>47.4.11</td>
<td>Register update</td>
<td>1784</td>
</tr>
<tr>
<td>47.4.12</td>
<td>Counter mode</td>
<td>1785</td>
</tr>
</tbody>
</table>
47.4.13 Timer enable ............................................. 1785
47.4.14 Timer counter reset ................................... 1786
47.4.15 Encoder mode ........................................... 1786
47.4.16 Debug mode .............................................. 1788
47.5 LPTIM low-power modes .................................. 1788
47.6 LPTIM interrupts ............................................ 1789
47.7 LPTIM registers ............................................. 1789
  47.7.1 LPTIM interrupt and status register (LPTIM_ISR) .......... 1790
  47.7.2 LPTIM interrupt clear register (LPTIM_ICR) .............. 1791
  47.7.3 LPTIM interrupt enable register (LPTIM_IER) ............. 1791
  47.7.4 LPTIM configuration register (LPTIM_CFGR) .............. 1792
  47.7.5 LPTIM control register (LPTIM_CR) ........................ 1795
  47.7.6 LPTIM compare register (LPTIM_CMP) ..................... 1796
  47.7.7 LPTIM autoreload register (LPTIM_ARR) .................. 1797
  47.7.8 LPTIM counter register (LPTIM_CNT) .................... 1797
  47.7.9 LPTIM configuration register 2 (LPTIM_CFGR2) .......... 1798
  47.7.10 LPTIM register map ................................... 1799

48 System window watchdog (WWDG) .......................... 1801
  48.1 Introduction .............................................. 1801
  48.2 WWDG main features ..................................... 1801
  48.3 WWDG functional description .............................. 1801
    48.3.1 WWDG block diagram ................................ 1802
    48.3.2 WWDG internal signals ............................... 1802
    48.3.3 Enabling the watchdog ................................ 1802
    48.3.4 Controlling the down-counter ......................... 1802
    48.3.5 How to program the watchdog timeout .................. 1803
    48.3.6 Debug mode ......................................... 1804
  48.4 WWDG interrupts ......................................... 1804
  48.5 WWDG registers .......................................... 1804
    48.5.1 WWDG control register (WWDG_CR) .................... 1804
    48.5.2 WWDG configuration register (WWDG_CFR) .............. 1805
    48.5.3 WWDG status register (WWDG_SR) ..................... 1806
    48.5.4 WWDG register map .................................. 1806

49 Independent watchdog (IWDG) ............................... 1807
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>49.1</td>
<td>Introduction</td>
<td>1807</td>
</tr>
<tr>
<td>49.2</td>
<td>IWDG main features</td>
<td>1807</td>
</tr>
<tr>
<td>49.3</td>
<td>IWDG functional description</td>
<td>1807</td>
</tr>
<tr>
<td>49.3.1</td>
<td>IWDG block diagram</td>
<td>1807</td>
</tr>
<tr>
<td>49.3.2</td>
<td>IWDG internal signals</td>
<td>1808</td>
</tr>
<tr>
<td>49.3.3</td>
<td>Window option</td>
<td>1808</td>
</tr>
<tr>
<td>49.3.4</td>
<td>Hardware watchdog</td>
<td>1809</td>
</tr>
<tr>
<td>49.3.5</td>
<td>Low-power freeze</td>
<td>1809</td>
</tr>
<tr>
<td>49.3.6</td>
<td>Register access protection</td>
<td>1809</td>
</tr>
<tr>
<td>49.3.7</td>
<td>Debug mode</td>
<td>1810</td>
</tr>
<tr>
<td>49.4</td>
<td>IWDG registers</td>
<td>1811</td>
</tr>
<tr>
<td>49.4.1</td>
<td>IWDG key register (IWDG_KR)</td>
<td>1811</td>
</tr>
<tr>
<td>49.4.2</td>
<td>IWDG prescaler register (IWDG_PR)</td>
<td>1812</td>
</tr>
<tr>
<td>49.4.3</td>
<td>IWDG reload register (IWDG_RLR)</td>
<td>1813</td>
</tr>
<tr>
<td>49.4.4</td>
<td>IWDG status register (IWDG_SR)</td>
<td>1814</td>
</tr>
<tr>
<td>49.4.5</td>
<td>IWDG window register (IWDG_WINR)</td>
<td>1815</td>
</tr>
<tr>
<td>49.4.6</td>
<td>IWDG register map</td>
<td>1816</td>
</tr>
<tr>
<td>50</td>
<td>Real-time clock (RTC)</td>
<td>1817</td>
</tr>
<tr>
<td>50.1</td>
<td>Introduction</td>
<td>1817</td>
</tr>
<tr>
<td>50.2</td>
<td>RTC main features</td>
<td>1817</td>
</tr>
<tr>
<td>50.3</td>
<td>RTC functional description</td>
<td>1818</td>
</tr>
<tr>
<td>50.3.1</td>
<td>RTC block diagram</td>
<td>1818</td>
</tr>
<tr>
<td>50.3.2</td>
<td>RTC pins and internal signals</td>
<td>1819</td>
</tr>
<tr>
<td>50.3.3</td>
<td>GPIOs controlled by the RTC and TAMP</td>
<td>1820</td>
</tr>
<tr>
<td>50.3.4</td>
<td>Clock and prescalers</td>
<td>1824</td>
</tr>
<tr>
<td>50.3.5</td>
<td>Real-time clock and calendar</td>
<td>1825</td>
</tr>
<tr>
<td>50.3.6</td>
<td>Programmable alarms</td>
<td>1825</td>
</tr>
<tr>
<td>50.3.7</td>
<td>Periodic auto-wakeup</td>
<td>1826</td>
</tr>
<tr>
<td>50.3.8</td>
<td>RTC initialization and configuration</td>
<td>1826</td>
</tr>
<tr>
<td>50.3.9</td>
<td>Reading the calendar</td>
<td>1828</td>
</tr>
<tr>
<td>50.3.10</td>
<td>Resetting the RTC</td>
<td>1829</td>
</tr>
<tr>
<td>50.3.11</td>
<td>RTC synchronization</td>
<td>1829</td>
</tr>
<tr>
<td>50.3.12</td>
<td>RTC reference clock detection</td>
<td>1830</td>
</tr>
<tr>
<td>50.3.13</td>
<td>RTC smooth digital calibration</td>
<td>1831</td>
</tr>
<tr>
<td>50.3.14</td>
<td>Timestamp function</td>
<td>1832</td>
</tr>
</tbody>
</table>
## 50.3.15 Calibration clock output ........................................ 1833
## 50.3.16 Tamper and alarm output .................................... 1834

### 50.4 RTC low-power modes ........................................... 1834

### 50.5 RTC interrupts .................................................. 1835

### 50.6 RTC registers .................................................... 1835

#### 50.6.1 RTC time register (RTC_TR) ................................. 1835
#### 50.6.2 RTC date register (RTC_DR) ................................. 1837
#### 50.6.3 RTC sub second register (RTC_SSR) ......................... 1838
#### 50.6.4 RTC initialization control and status register (RTC_ICSR) ........................................... 1838
#### 50.6.5 RTC prescaler register (RTC_PRER) ......................... 1840
#### 50.6.6 RTC wakeup timer register (RTC_WUTR) ..................... 1840
#### 50.6.7 RTC control register (RTC_CR) .............................. 1841
#### 50.6.8 RTC write protection register (RTC_WPR) .................. 1844
#### 50.6.9 RTC calibration register (RTC_CALR) ....................... 1845
#### 50.6.10 RTC shift control register (RTC_SHIFTR) ................ 1846
#### 50.6.11 RTC timestamp time register (RTC_TSTR) ............... 1847
#### 50.6.12 RTC timestamp date register (RTC_TSDR) ................ 1847
#### 50.6.13 RTC timestamp sub second register (RTC_TSSSR) ........ 1848
#### 50.6.14 RTC alarm A register (RTC_ALRMAR) ...................... 1849
#### 50.6.15 RTC alarm A sub second register (RTC_ALRMASSR) ...... 1850
#### 50.6.16 RTC alarm B register (RTC_ALRMBR) ....................... 1851
#### 50.6.17 RTC alarm B sub second register (RTC_ALRMBSSR) ...... 1852
#### 50.6.18 RTC status register (RTC_SR) ............................. 1852
#### 50.6.19 RTC masked interrupt status register (RTC_MISR) ...... 1853
#### 50.6.20 RTC status clear register (RTC_SCR) ..................... 1854
#### 50.6.21 RTC configuration register (RTC_CFGR) .................. 1855
#### 50.6.22 RTC register map ............................................ 1856

## 51 Tamper and backup registers (TAMP) ............................ 1858

### 51.1 Introduction .................................................... 1858

### 51.2 TAMP main features ............................................ 1858

### 51.3 TAMP functional description .................................. 1859

#### 51.3.1 TAMP block diagram ....................................... 1859
#### 51.3.2 TAMP pins and internal signals ............................ 1860
#### 51.3.3 TAMP register write protection ............................ 1861
#### 51.3.4 Tamper detection ............................................ 1861
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>51.4</td>
<td>TAMP low-power modes</td>
<td>1865</td>
</tr>
<tr>
<td>51.5</td>
<td>TAMP interrupts</td>
<td>1865</td>
</tr>
<tr>
<td>51.6</td>
<td>TAMP registers</td>
<td>1865</td>
</tr>
<tr>
<td>51.6.1</td>
<td>TAMP control register 1 (TAMP_CR1)</td>
<td>1866</td>
</tr>
<tr>
<td>51.6.2</td>
<td>TAMP control register 2 (TAMP_CR2)</td>
<td>1867</td>
</tr>
<tr>
<td>51.6.3</td>
<td>TAMP filter control register (TAMPFLTCR)</td>
<td>1868</td>
</tr>
<tr>
<td>51.6.4</td>
<td>TAMP active tamper control register 1 (TAMP_ATCR1)</td>
<td>1869</td>
</tr>
<tr>
<td>51.6.5</td>
<td>TAMP active tamper seed register (TAMP_ATSEEDR)</td>
<td>1871</td>
</tr>
<tr>
<td>51.6.6</td>
<td>TAMP active tamper output register (TAMP_ATOR)</td>
<td>1872</td>
</tr>
<tr>
<td>51.6.7</td>
<td>TAMP interrupt enable register (TAMP_IER)</td>
<td>1872</td>
</tr>
<tr>
<td>51.6.8</td>
<td>TAMP status register (TAMP_SR)</td>
<td>1874</td>
</tr>
<tr>
<td>51.6.9</td>
<td>TAMP masked interrupt status register (TAMP_MISR)</td>
<td>1875</td>
</tr>
<tr>
<td>51.6.10</td>
<td>TAMP status clear register (TAMP_SCR)</td>
<td>1876</td>
</tr>
<tr>
<td>51.6.11</td>
<td>TAMP monotonic counter register (TAMP_COUNTR)</td>
<td>1877</td>
</tr>
<tr>
<td>51.6.12</td>
<td>TAMP configuration register (TAMP_CFGR)</td>
<td>1877</td>
</tr>
<tr>
<td>51.6.13</td>
<td>TAMP backup x register (TAMP_BKPxR)</td>
<td>1878</td>
</tr>
<tr>
<td>51.6.14</td>
<td>TAMP register map</td>
<td>1879</td>
</tr>
</tbody>
</table>

### 52 Inter-integrated circuit (I2C) interface

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>52.1</td>
<td>Introduction</td>
<td>1881</td>
</tr>
<tr>
<td>52.2</td>
<td>I2C main features</td>
<td>1881</td>
</tr>
<tr>
<td>52.3</td>
<td>I2C implementation</td>
<td>1882</td>
</tr>
<tr>
<td>52.4</td>
<td>I2C functional description</td>
<td>1882</td>
</tr>
<tr>
<td>52.4.1</td>
<td>I2C block diagram</td>
<td>1883</td>
</tr>
<tr>
<td>52.4.2</td>
<td>I2C pins and internal signals</td>
<td>1884</td>
</tr>
<tr>
<td>52.4.3</td>
<td>I2C clock requirements</td>
<td>1884</td>
</tr>
<tr>
<td>52.4.4</td>
<td>Mode selection</td>
<td>1884</td>
</tr>
<tr>
<td>52.4.5</td>
<td>I2C initialization</td>
<td>1885</td>
</tr>
<tr>
<td>52.4.6</td>
<td>Software reset</td>
<td>1890</td>
</tr>
<tr>
<td>52.4.7</td>
<td>Data transfer</td>
<td>1891</td>
</tr>
<tr>
<td>52.4.8</td>
<td>I2C slave mode</td>
<td>1893</td>
</tr>
<tr>
<td>52.4.9</td>
<td>I2C master mode</td>
<td>1902</td>
</tr>
<tr>
<td>52.4.10</td>
<td>I2C_TIMINGR register configuration examples</td>
<td>1914</td>
</tr>
<tr>
<td>52.4.11</td>
<td>SMBus specific features</td>
<td>1915</td>
</tr>
<tr>
<td>52.4.12</td>
<td>SMBus initialization</td>
<td>1918</td>
</tr>
<tr>
<td>52.4.13</td>
<td>SMBus: I2C_TIMEOUTR register configuration examples</td>
<td>1920</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>52.4.14 SMBus slave mode</td>
<td>1921</td>
<td></td>
</tr>
<tr>
<td>52.4.15 Wakeup from Stop mode on address match</td>
<td>1929</td>
<td></td>
</tr>
<tr>
<td>52.4.16 Error conditions</td>
<td>1929</td>
<td></td>
</tr>
<tr>
<td>52.4.17 DMA requests</td>
<td>1931</td>
<td></td>
</tr>
<tr>
<td>52.4.18 Debug mode</td>
<td>1932</td>
<td></td>
</tr>
<tr>
<td>52.5 I2C low-power modes</td>
<td>1932</td>
<td></td>
</tr>
<tr>
<td>52.6 I2C interrupts</td>
<td>1933</td>
<td></td>
</tr>
<tr>
<td>52.7 I2C registers</td>
<td>1934</td>
<td></td>
</tr>
<tr>
<td>52.7.1 I2C control register 1 (I2C_CR1)</td>
<td>1934</td>
<td></td>
</tr>
<tr>
<td>52.7.2 I2C control register 2 (I2C_CR2)</td>
<td>1937</td>
<td></td>
</tr>
<tr>
<td>52.7.3 I2C own address 1 register (I2C_OAR1)</td>
<td>1940</td>
<td></td>
</tr>
<tr>
<td>52.7.4 I2C own address 2 register (I2C_OAR2)</td>
<td>1941</td>
<td></td>
</tr>
<tr>
<td>52.7.5 I2C timing register (I2C_TIMINGR)</td>
<td>1942</td>
<td></td>
</tr>
<tr>
<td>52.7.6 I2C timeout register (I2C_TIMEOUTR)</td>
<td>1943</td>
<td></td>
</tr>
<tr>
<td>52.7.7 I2C interrupt and status register (I2C_ISR)</td>
<td>1944</td>
<td></td>
</tr>
<tr>
<td>52.7.8 I2C interrupt clear register (I2C_ICR)</td>
<td>1946</td>
<td></td>
</tr>
<tr>
<td>52.7.9 I2C PEC register (I2C_PECR)</td>
<td>1947</td>
<td></td>
</tr>
<tr>
<td>52.7.10 I2C receive data register (I2C_RXDR)</td>
<td>1948</td>
<td></td>
</tr>
<tr>
<td>52.7.11 I2C transmit data register (I2C_TXDR)</td>
<td>1948</td>
<td></td>
</tr>
<tr>
<td>52.7.12 I2C register map</td>
<td>1949</td>
<td></td>
</tr>
<tr>
<td>53 Universal synchronous/asynchronous receiver transmitter (USART/UART)</td>
<td>1951</td>
<td></td>
</tr>
<tr>
<td>53.1 USART introduction</td>
<td>1951</td>
<td></td>
</tr>
<tr>
<td>53.2 USART main features</td>
<td>1952</td>
<td></td>
</tr>
<tr>
<td>53.3 USART extended features</td>
<td>1953</td>
<td></td>
</tr>
<tr>
<td>53.4 USART implementation</td>
<td>1953</td>
<td></td>
</tr>
<tr>
<td>53.5 USART functional description</td>
<td>1954</td>
<td></td>
</tr>
<tr>
<td>53.5.1 USART block diagram</td>
<td>1954</td>
<td></td>
</tr>
<tr>
<td>53.5.2 USART signals</td>
<td>1955</td>
<td></td>
</tr>
<tr>
<td>53.5.3 USART character description</td>
<td>1956</td>
<td></td>
</tr>
<tr>
<td>53.5.4 USART FIFOs and thresholds</td>
<td>1958</td>
<td></td>
</tr>
<tr>
<td>53.5.5 USART transmitter</td>
<td>1958</td>
<td></td>
</tr>
<tr>
<td>53.5.6 USART receiver</td>
<td>1962</td>
<td></td>
</tr>
<tr>
<td>53.5.7 USART baud rate generation</td>
<td>1969</td>
<td></td>
</tr>
<tr>
<td>53.5.8 Tolerance of the USART receiver to clock deviation</td>
<td>1970</td>
<td></td>
</tr>
</tbody>
</table>
53.5.9 USART Auto baud rate detection ........................................ 1972
53.5.10 USART multiprocessor communication ............................. 1974
53.5.11 USART Modbus communication ..................................... 1976
53.5.12 USART parity control .................................................. 1976
53.5.13 USART LIN (local interconnection network) mode ............. 1977
53.5.14 USART synchronous mode .............................................. 1980
53.5.15 USART single-wire Half-duplex communication ................. 1984
53.5.16 USART receiver timeout ............................................... 1984
53.5.17 USART Smartcard mode ............................................... 1985
53.5.18 USART IrDA SIR ENDEC block ....................................... 1989
53.5.19 Continuous communication using USART and DMA .......... 1992
53.5.20 RS232 Hardware flow control and RS485 Driver Enable ...... 1994
53.5.21 USART low-power management ..................................... 1997

53.6 USART interrupts .......................................................... 2000

53.7 USART registers ............................................................ 2003
53.7.1 USART control register 1 [alternate] (USART_CR1) ............ 2003
53.7.2 USART control register 1 [alternate] (USART_CR1) ............ 2007
53.7.3 USART control register 2 (USART_CR2) ........................... 2010
53.7.4 USART control register 3 (USART_CR3) ........................... 2014
53.7.5 USART baud rate register (USART_BRR) .......................... 2019
53.7.6 USART guard time and prescaler register (USART_GTPR) ...... 2019
53.7.7 USART receiver timeout register (USART_RTOR) .............. 2020
53.7.8 USART request register (USART_RQR) .............................. 2021
53.7.9 USART interrupt and status register [alternate] (USART_ISR) .. 2022
53.7.10 USART interrupt and status register [alternate] (USART_ISR) .. 2028
53.7.11 USART interrupt flag clear register (USART_ICR) ............. 2033
53.7.12 USART receive data register (USART_RDR) ...................... 2035
53.7.13 USART transmit data register (USART_TDR) .................... 2035
53.7.14 USART prescaler register (USART_PRES) ....................... 2036
53.7.15 USART register map ................................................... 2037

54 Low-power universal asynchronous receiver transmitter (LPUART) .................................. 2039
54.1 LPUART introduction ....................................................... 2039
54.2 LPUART main features .................................................... 2040
54.3 LPUART implementation .................................................. 2040
54.4 LPUART functional description .......................................... 2042
### Contents

54.4.1 LPUART block diagram ................................................. 2042
54.4.2 LPUART signals ......................................................... 2043
54.4.3 LPUART character description ......................................... 2043
54.4.4 LPUART FIFOs and thresholds ........................................ 2044
54.4.5 LPUART transmitter ..................................................... 2045
54.4.6 LPUART receiver ......................................................... 2048
54.4.7 LPUART baud rate generation ......................................... 2052
54.4.8 Tolerance of the LPUART receiver to clock deviation ............... 2053
54.4.9 LPUART multiprocessor communication ............................... 2054
54.4.10 LPUART parity control ................................................ 2056
54.4.11 LPUART single-wire Half-duplex communication .................... 2057
54.4.12 Continuous communication using DMA and LPUART ................. 2057
54.4.13 RS232 Hardware flow control and RS485 Driver Enable ............. 2060
54.4.14 LPUART low-power management ...................................... 2062

54.5 LPUART interrupts ......................................................... 2065

54.6 LPUART registers .......................................................... 2067
54.6.1 LPUART control register 1 [alternate] (LPUART_CR1) ................. 2067
54.6.2 LPUART control register 1 [alternate] (LPUART_CR1) ................. 2070
54.6.3 LPUART control register 2 (LPUART_CR2) ............................. 2073
54.6.4 LPUART control register 3 (LPUART_CR3) ............................. 2075
54.6.5 LPUART baud rate register (LPUART_BRR) ........................... 2078
54.6.6 LPUART request register (LPUART_RQR) ............................. 2079
54.6.7 LPUART interrupt and status register [alternate] (LPUART_ISR) ... 2079
54.6.8 LPUART interrupt and status register [alternate] (LPUART_ISR) ... 2084
54.6.9 LPUART interrupt flag clear register (LPUART_ICR) .................. 2087
54.6.10 LPUART receive data register (LPUART_RDR) ....................... 2088
54.6.11 LPUART transmit data register (LPUART_TDR) ...................... 2088
54.6.12 LPUART prescaler register (LPUART_PRESC) ........................ 2089
54.6.13 LPUART register map .................................................. 2090

55 Serial peripheral interface (SPI) ............................................ 2092
55.1 Introduction ..................................................................... 2092
55.2 SPI main features ............................................................ 2093
55.3 SPI implementation .......................................................... 2093
55.4 SPI functional description ................................................ 2094
55.4.1 SPI block diagram ....................................................... 2094
55.4.2 SPI signals ........................................... 2095
55.4.3 SPI communication general aspects .................. 2095
55.4.4 Communications between one master and one slave ........................................... 2095
55.4.5 Standard multi-slave communication .................. 2098
55.4.6 Multi-master communication .......................... 2101
55.4.7 Slave select (SS) pin management .................... 2101
55.4.8 Communication formats ................................ 2105
55.4.9 Configuration of SPI .................................. 2107
55.4.10 Procedure for enabling SPI ........................... 2108
55.4.11 SPI data transmission and reception procedures ........ 2108
55.4.12 Procedure for disabling the SPI ....................... 2112
55.4.13 Data packing ........................................ 2113
55.4.14 Communication using DMA (direct memory addressing) .................. 2114
55.5 SPI specific modes and control ........................ 2116
55.5.1 TI mode ............................................. 2116
55.5.2 SPI error flags ..................................... 2116
55.5.3 CRC computation ..................................... 2118
55.6 Low-power mode management ............................ 2120
55.7 SPI wakeup and interrupts ............................... 2122
55.8 I2S main features ...................................... 2123
55.9 I2S functional description ............................... 2124
55.9.1 I2S general description ................................ 2124
55.9.2 Pin sharing with SPI function ........................ 2124
55.9.3 Bits and fields usable in I2S/PCM mode ............... 2125
55.9.4 Slave and master modes ............................... 2126
55.9.5 Supported audio protocols .......................... 2126
55.9.6 Additional Serial Interface Flexibility ................ 2132
55.9.7 Start-up sequence ................................... 2134
55.9.8 Stop sequence ....................................... 2135
55.9.9 Clock generator ..................................... 2136
55.9.10 Internal FIFOs ...................................... 2138
55.9.11 FIFOs status flags ................................. 2139
55.9.12 Handling of underrun situation ....................... 2140
55.9.13 Handling of overrun situation ....................... 2141
55.9.14 Frame error detection ................................ 2141
55.9.15 DMA Interface ..................................... 2143
## 55

Programing examples ........................................... 2144
Slave I2S Philips standard, receive ............................ 2146
I2S wakeup and interrupts ....................................... 2147

### 55.11 SPI/I2S registers ................................. 2148

#### 55.11.1 SPI/I2S control register 1 (SPI2S_CR1) ............ 2148
#### 55.11.2 SPI control register 2 (SPI_CR2) .................. 2149
#### 55.11.3 SPI configuration register 1 (SPI_CFG1) .......... 2150
#### 55.11.4 SPI configuration register 2 (SPI_CFG2) .......... 2153
#### 55.11.5 SPI/I2S interrupt enable register (SPI2S_IER) .... 2155
#### 55.11.6 SPI/I2S status register (SPI2S_SR) ............... 2156
#### 55.11.7 SPI/I2S interrupt/status flags clear register (SPI2S_IFCR) .... 2158
#### 55.11.8 SPI/I2S transmit data register (SPI2S_TXDR) ...... 2159
#### 55.11.9 SPI/I2S receive data register (SPI2S_RXDR) ...... 2160
#### 55.11.10 SPI polynomial register (SPI_CRCPOLY) .......... 2160
#### 55.11.11 SPI transmitter CRC register (SPI_TXCRC) ....... 2161
#### 55.11.12 SPI receiver CRC register (SPI_RXCRC) ........ 2161
#### 55.11.13 SPI underrun data register (SPI_UDRDR) ......... 2162
#### 55.11.14 SPI/I2S configuration register (SPI_I2SCFGR) .... 2162

### 55.12 SPI register map and reset values .................. 2165

## 56 Serial audio interface (SAI) ......................... 2167

### 56.1 Introduction ........................................ 2167
### 56.2 SAI main features ................................... 2168
### 56.3 SAI implementation .................................. 2168
### 56.4 SAI functional description .......................... 2169

#### 56.4.1 SAI block diagram ................................ 2169
#### 56.4.2 SAI pins and internal signals .................... 2170
#### 56.4.3 Main SAI modes ................................... 2171
#### 56.4.4 SAI synchronization mode ......................... 2172
#### 56.4.5 Audio data size .................................. 2173
#### 56.4.6 Frame synchronization ............................ 2173
#### 56.4.7 Slot configuration ................................ 2176
#### 56.4.8 SAI clock generator .............................. 2178
#### 56.4.9 Internal FIFOs .................................... 2181
#### 56.4.10 PDM Interface ................................... 2183
#### 56.4.11 AC’97 link controller ............................ 2191
56.4.12 SPDIF output ........................................... 2193
56.4.13 Specific features ...................................... 2196
56.4.14 Error flags ............................................. 2200
56.4.15 Disabling the SAI ...................................... 2203
56.4.16 SAI DMA interface .................................... 2203
56.5 SAI interrupts ............................................... 2204
56.6 SAI registers ................................................. 2206
  56.6.1 SAI global configuration register (SAI_GCR) ............. 2206
  56.6.2 SAI configuration register 1 (SAI_ACR1) ................. 2206
  56.6.3 SAI configuration register 1 (SAI_BCR1) ................. 2209
  56.6.4 SAI configuration register 2 (SAI_ACR2) ................. 2212
  56.6.5 SAI configuration register 2 (SAI_BCR2) ................. 2214
  56.6.6 SAI frame configuration register (SAI_AFRCR) ............ 2216
  56.6.7 SAI frame configuration register (SAI_BFRCR) ............ 2217
  56.6.8 SAI slot register (SAI_A SLOT R) ....................... 2218
  56.6.9 SAI slot register (SAI_B SLOT R) ....................... 2219
  56.6.10 SAI interrupt mask register (SAI_AIM) .................. 2220
  56.6.11 SAI interrupt mask register (SAI_BIM) .................. 2222
  56.6.12 SAI status register (SAI_ASR) ........................ 2223
  56.6.13 SAI status register (SAI_BSR) ........................ 2225
  56.6.14 SAI clear flag register (SAI_ACLRFR) ................... 2227
  56.6.15 SAI clear flag register (SAI_BCLRFR) ................... 2228
  56.6.16 SAI data register (SAI_ADR) ........................ 2229
  56.6.17 SAI data register (SAI_BDR) ........................ 2230
  56.6.18 SAI PDM control register (SAI_PDMCR) .................. 2230
  56.6.19 SAI PDM delay register (SAI_PDMDLY) ............... 2231
  56.6.20 SAI register map ..................................... 2234
57  SPDIF receiver interface (SPDIFRX) .......................... 2236
  57.1 SPDIFRX interface introduction ............................ 2236
  57.2 SPDIFRX main features .................................... 2236
  57.3 SPDIFRX functional description ............................ 2236
    57.3.1 SPDIFRX pins and internal signals ...................... 2237
    57.3.2 SPDIF protocol (IEC-60958) .......................... 2238
    57.3.3 SPDIFRX decoder (SPDIFRX_DC) ....................... 2240
    57.3.4 SPDIFRX tolerance to clock deviation ................. 2244
    57.3.5 SPDIFRX synchronization ............................. 2244
57.3.6 SPDIFRX handling ............................................. 2246
57.3.7 Data reception management ................................. 2248
57.3.8 Dedicated control flow ...................................... 2250
57.3.9 Reception errors ............................................. 2251
57.3.10 Clocking strategy ........................................... 2253
57.3.11 Symbol clock generation ................................... 2254
57.3.12 DMA Interface ............................................... 2255
57.3.13 Interrupt Generation ........................................ 2256
57.3.14 Register protection .......................................... 2257

57.4 Programming procedures ...................................... 2257
57.4.1 Initialization phase ........................................... 2258
57.4.2 Handling of interrupts coming from SPDIFRX ........ 2259
57.4.3 Handling of interrupts coming from DMA .......... 2259

57.5 SPDIFRX interface registers ................................. 2260
57.5.1 Control register (SPDIFRX_CR) .............................. 2260
57.5.2 Interrupt mask register (SPDIFRX_IMR) .................... 2263
57.5.3 Status register (SPDIFRX_SR) ................................. 2264
57.5.4 Interrupt flag clear register (SPDIFRX_IFCR) .......... 2266
57.5.5 Data input register (SPDIFRX_FMT0_DR) .............. 2267
57.5.6 Data input register (SPDIFRX_FMT1_DR) .............. 2268
57.5.7 Data input register (SPDIFRX_FMT2_DR) .............. 2269
57.5.8 Channel status register (SPDIFRX_CSR) ............ 2270
57.5.9 Debug information register (SPDIFRX_DIR) ........ 2270
57.5.10 SPDIFRX interface register map ......................... 2272

58 Single Wire Protocol Master Interface (SWPMI) ............ 2273
58.1 Introduction ...................................................... 2273
58.2 SWPMI main features ........................................... 2274
58.3 SWPMI functional description ................................. 2275
58.3.1 SWPMI block diagram ....................................... 2275
58.3.2 SWPMI pins and internal signals .......................... 2275
58.3.3 SWP initialization and activation ....................... 2276
58.3.4 SWP bus states ............................................... 2277
58.3.5 SWPMI_IO (internal transceiver) bypass ............ 2278
58.3.6 SWPMI Bit rate ............................................... 2278
58.3.7 SWPMI frame handling ..................................... 2279
58.3.8 Transmission procedure .................................... 2279
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>58.3.9</td>
<td>Reception procedure</td>
<td>2284</td>
</tr>
<tr>
<td>58.3.10</td>
<td>Error management</td>
<td>2288</td>
</tr>
<tr>
<td>58.3.11</td>
<td>Loopback mode</td>
<td>2290</td>
</tr>
<tr>
<td>58.4</td>
<td>SWPMI low-power modes</td>
<td>2290</td>
</tr>
<tr>
<td>58.5</td>
<td>SWPMI interrupts</td>
<td>2291</td>
</tr>
<tr>
<td>58.6</td>
<td>SWPMI registers</td>
<td>2292</td>
</tr>
<tr>
<td>58.6.1</td>
<td>SWPMI Configuration/Control register (SWPMI_CR)</td>
<td>2292</td>
</tr>
<tr>
<td>58.6.2</td>
<td>SWPMI Bitrate register (SWPMI_BRR)</td>
<td>2293</td>
</tr>
<tr>
<td>58.6.3</td>
<td>SWPMI Interrupt and Status register (SWPMI_ISR)</td>
<td>2294</td>
</tr>
<tr>
<td>58.6.4</td>
<td>SWPMI Interrupt Flag Clear register (SWPMI_ICR)</td>
<td>2295</td>
</tr>
<tr>
<td>58.6.5</td>
<td>SWPMI Interrupt Enable register (SWPMI_IER)</td>
<td>2296</td>
</tr>
<tr>
<td>58.6.6</td>
<td>SWPMI Receive Frame Length register (SWPMI_RFL)</td>
<td>2298</td>
</tr>
<tr>
<td>58.6.7</td>
<td>SWPMI Transmit data register (SWPMI_TDR)</td>
<td>2298</td>
</tr>
<tr>
<td>58.6.8</td>
<td>SWPMI Receive data register (SWPMI_RDR)</td>
<td>2298</td>
</tr>
<tr>
<td>58.6.9</td>
<td>SWPMI Option register (SWPMI_OR)</td>
<td>2299</td>
</tr>
<tr>
<td>58.6.10</td>
<td>SWPMI register map and reset value table</td>
<td>2300</td>
</tr>
<tr>
<td>59</td>
<td>Management data input/output (MDIOS)</td>
<td>2301</td>
</tr>
<tr>
<td>59.1</td>
<td>MDIOS introduction</td>
<td>2301</td>
</tr>
<tr>
<td>59.2</td>
<td>MDIOS main features</td>
<td>2301</td>
</tr>
<tr>
<td>59.3</td>
<td>MDIOS functional description</td>
<td>2302</td>
</tr>
<tr>
<td>59.3.1</td>
<td>MDIOS block diagram</td>
<td>2302</td>
</tr>
<tr>
<td>59.3.2</td>
<td>MDIOS pins and internal signals</td>
<td>2302</td>
</tr>
<tr>
<td>59.3.3</td>
<td>MDIOS protocol</td>
<td>2303</td>
</tr>
<tr>
<td>59.3.4</td>
<td>MDIOS enabling and disabling</td>
<td>2304</td>
</tr>
<tr>
<td>59.3.5</td>
<td>MDIOS data</td>
<td>2304</td>
</tr>
<tr>
<td>59.3.6</td>
<td>MDIOS APB frequency</td>
<td>2306</td>
</tr>
<tr>
<td>59.3.7</td>
<td>Write/read flags and interrupts</td>
<td>2306</td>
</tr>
<tr>
<td>59.3.8</td>
<td>MDIOS error management</td>
<td>2306</td>
</tr>
<tr>
<td>59.3.9</td>
<td>MDIOS in Stop mode</td>
<td>2307</td>
</tr>
<tr>
<td>59.3.10</td>
<td>MDIOS interrupts</td>
<td>2307</td>
</tr>
<tr>
<td>59.4</td>
<td>MDIOS registers</td>
<td>2308</td>
</tr>
<tr>
<td>59.4.1</td>
<td>MDIOS configuration register (MDIOS_CR)</td>
<td>2308</td>
</tr>
<tr>
<td>59.4.2</td>
<td>MDIOS write flag register (MDIOS_WRFR)</td>
<td>2309</td>
</tr>
<tr>
<td>59.4.3</td>
<td>MDIOS clear write flag register (MDIOS_CWRFR)</td>
<td>2309</td>
</tr>
<tr>
<td>59.4.4</td>
<td>MDIOS read flag register (MDIOS_RDFR)</td>
<td>2310</td>
</tr>
</tbody>
</table>
60 Secure digital input/output MultiMediaCard interface (SDMMC) ........................................ 2315
60.1 SDMMC main features ........................................ 2315
60.2 SDMMC implementation ........................................ 2315
60.3 SDMMC bus topology ........................................ 2316
60.4 SDMMC operation modes ........................................ 2318
60.5 SDMMC functional description ........................................ 2319
  60.5.1 SDMMC block diagram ........................................ 2319
  60.5.2 SDMMC pins and internal signals ........................................ 2319
  60.5.3 General description ........................................ 2320
  60.5.4 SDMMC adapter ........................................ 2322
  60.5.5 SDMMC AHB slave interface ........................................ 2343
  60.5.6 SDMMC AHB master interface ........................................ 2344
  60.5.7 MDMA request generation ........................................ 2346
  60.5.8 AHB and SDMMC_CK clock relation ........................................ 2346
  60.5.9 Hardware flow control ........................................ 2347
60.6 Card functional description ........................................ 2348
  60.6.1 SD I/O mode ........................................ 2348
  60.6.2 CMD12 send timing ........................................ 2356
  60.6.3 Sleep (CMD5) ........................................ 2359
  60.6.4 Interrupt mode (Wait-IRQ) ........................................ 2360
  60.6.5 Boot operation ........................................ 2361
  60.6.6 Response R1b handling ........................................ 2364
  60.6.7 Reset and card cycle power ........................................ 2365
60.7 Ultra-high-speed phase I (UHS-I) voltage switch ........................................ 2366
60.8 SDMMC interrupts ........................................ 2370
60.9 SDMMC registers ........................................ 2372
  60.9.1 SDMMC power control register (SDMMC_POWER) ........................................ 2372
  60.9.2 SDMMC clock control register (SDMMC_CLKCR) ........................................ 2373
  60.9.3 SDMMC argument register (SDMMC_ARGR) ........................................ 2375
60.9.4 SDMMC command register (SDMMC_CMDR) .......................... 2375
60.9.5 SDMMC command response register (SDMMC_RESPCMDR) ... 2377
60.9.6 SDMMC response x register (SDMMC_RESPxR) .................... 2378
60.9.7 SDMMC data timer register (SDMMC_DTIMER) .................... 2378
60.9.8 SDMMC data length register (SDMMC_DLENR) .................... 2379
60.9.9 SDMMC data control register (SDMMC_DCTRL) ................... 2380
60.9.10 SDMMC data counter register (SDMMC_DCNTR) ................. 2381
60.9.11 SDMMC status register (SDMMC_STAR) .......................... 2382
60.9.12 SDMMC interrupt clear register (SDMMC_ICR) ................. 2385
60.9.13 SDMMC mask register (SDMMC_MASKR) .......................... 2387
60.9.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER) . 2390
60.9.15 SDMMC data FIFO registers x (SDMMC_FIFORx) ............... 2390
60.9.16 SDMMC DMA control register (SDMMC_IDMACTRLR) .......... 2391
60.9.17 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER) .... 2392
60.9.18 SDMMC IDMA buffer 0 base address register (SDMMC_IDMABASE0R) .................................................. 2392
60.9.19 SDMMC IDMA buffer 1 base address register (SDMMC_IDMABASE1R) .................................................. 2393
60.9.20 SDMMC register map ................................................. 2394

61 Controller area network with flexible data rate (FDCAN) ............ 2397
61.1 Introduction .......................................................... 2397
61.2 FDCAN main features ................................................. 2400
61.3 FDCAN implementation ............................................... 2400
61.4 FDCAN functional description ...................................... 2401
  61.4.1 Operating modes .................................................. 2402
  61.4.2 Message RAM ..................................................... 2411
  61.4.3 FIFO acknowledge handling ................................... 2422
  61.4.4 Bit timing ......................................................... 2423
  61.4.5 Clock calibration on CAN ....................................... 2424
  61.4.6 Application ....................................................... 2428
  61.4.7 TTCAN operations (FDCAN1 only) ............................................. 2429
  61.4.8 TTCAN configuration ............................................ 2430
  61.4.9 Message scheduling ............................................ 2432
  61.4.10 TTCAN gap control ........................................... 2439
  61.4.11 Stop watch ...................................................... 2440
  61.4.12 Local time, cycle time, global time,
           and external clock synchronization .......................... 2440
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>61.4.13</td>
<td>TTCAN error level</td>
<td>2443</td>
</tr>
<tr>
<td>61.4.14</td>
<td>TTCAN message handling</td>
<td>2444</td>
</tr>
<tr>
<td>61.4.15</td>
<td>TTCAN interrupt and error handling</td>
<td>2447</td>
</tr>
<tr>
<td>61.4.16</td>
<td>Level 0</td>
<td>2448</td>
</tr>
<tr>
<td>61.4.17</td>
<td>Synchronization to external time schedule</td>
<td>2450</td>
</tr>
<tr>
<td>61.4.18</td>
<td>FDCAN Rx buffer and FIFO element</td>
<td>2451</td>
</tr>
<tr>
<td>61.4.19</td>
<td>FDCAN Tx buffer element</td>
<td>2453</td>
</tr>
<tr>
<td>61.4.20</td>
<td>FDCAN Tx event FIFO element</td>
<td>2455</td>
</tr>
<tr>
<td>61.4.21</td>
<td>FDCAN standard message ID filter element</td>
<td>2456</td>
</tr>
<tr>
<td>61.4.22</td>
<td>FDCAN extended message ID filter element</td>
<td>2458</td>
</tr>
<tr>
<td>61.4.23</td>
<td>FDCAN trigger memory element</td>
<td>2459</td>
</tr>
<tr>
<td>61.5</td>
<td>FDCAN registers</td>
<td>2461</td>
</tr>
<tr>
<td>61.5.1</td>
<td>FDCAN core release register (FDCAN_CREL)</td>
<td>2461</td>
</tr>
<tr>
<td>61.5.2</td>
<td>FDCAN Endian register (FDCAN_ENDIAN)</td>
<td>2461</td>
</tr>
<tr>
<td>61.5.3</td>
<td>FDCAN data bit timing and prescaler register (FDCAN_DBTP)</td>
<td>2461</td>
</tr>
<tr>
<td>61.5.4</td>
<td>FDCAN test register (FDCAN_TEST)</td>
<td>2462</td>
</tr>
<tr>
<td>61.5.5</td>
<td>FDCAN RAM watchdog register (FDCAN_RWD)</td>
<td>2463</td>
</tr>
<tr>
<td>61.5.6</td>
<td>FDCAN CC control register (FDCAN_CCCR)</td>
<td>2464</td>
</tr>
<tr>
<td>61.5.7</td>
<td>FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)</td>
<td>2466</td>
</tr>
<tr>
<td>61.5.8</td>
<td>FDCAN timestamp counter configuration register (FDCAN_TSCC)</td>
<td>2467</td>
</tr>
<tr>
<td>61.5.9</td>
<td>FDCAN timestamp counter value register (FDCAN_TSCV)</td>
<td>2467</td>
</tr>
<tr>
<td>61.5.10</td>
<td>FDCAN timeout counter configuration register (FDCAN_TOCC)</td>
<td>2468</td>
</tr>
<tr>
<td>61.5.11</td>
<td>FDCAN timeout counter value register (FDCAN_TOCV)</td>
<td>2469</td>
</tr>
<tr>
<td>61.5.12</td>
<td>FDCAN error counter register (FDCAN_ECR)</td>
<td>2469</td>
</tr>
<tr>
<td>61.5.13</td>
<td>FDCAN protocol status register (FDCAN_PSR)</td>
<td>2470</td>
</tr>
<tr>
<td>61.5.14</td>
<td>FDCAN transmitter delay compensation register (FDCAN_TDCR)</td>
<td>2472</td>
</tr>
<tr>
<td>61.5.15</td>
<td>FDCAN interrupt register (FDCAN_IR)</td>
<td>2473</td>
</tr>
<tr>
<td>61.5.16</td>
<td>FDCAN interrupt enable register (FDCAN_IE)</td>
<td>2476</td>
</tr>
<tr>
<td>61.5.17</td>
<td>FDCAN interrupt line select register (FDCAN_ILE)</td>
<td>2478</td>
</tr>
<tr>
<td>61.5.18</td>
<td>FDCAN interrupt line enable register (FDCAN_ILE)</td>
<td>2479</td>
</tr>
<tr>
<td>61.5.19</td>
<td>FDCAN global filter configuration register (FDCAN_GFC)</td>
<td>2480</td>
</tr>
<tr>
<td>61.5.20</td>
<td>FDCAN standard ID filter configuration register (FDCAN_SIDFC)</td>
<td>2481</td>
</tr>
<tr>
<td>61.5.21</td>
<td>FDCAN extended ID filter configuration register (FDCAN_XIDFC)</td>
<td>2481</td>
</tr>
<tr>
<td>61.5.22</td>
<td>FDCAN extended ID and mask register (FDCAN_XIDAM)</td>
<td>2482</td>
</tr>
<tr>
<td>61.5.23</td>
<td>FDCAN high priority message status register (FDCAN_HPMS)</td>
<td>2483</td>
</tr>
<tr>
<td>61.5.24</td>
<td>FDCAN new data 1 register (FDCAN_NDAT1)</td>
<td>2483</td>
</tr>
<tr>
<td>61.5.25</td>
<td>FDCAN new data 2 register (FDCAN_NDAT2)</td>
<td>2484</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>61.5.26</td>
<td>FDCAN Rx FIFO 0 configuration register (FDCAN_RXF0C)</td>
<td></td>
</tr>
<tr>
<td>61.5.27</td>
<td>FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)</td>
<td></td>
</tr>
<tr>
<td>61.5.28</td>
<td>FDCAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)</td>
<td></td>
</tr>
<tr>
<td>61.5.29</td>
<td>FDCAN Rx buffer configuration register (FDCAN_RXBC)</td>
<td></td>
</tr>
<tr>
<td>61.5.30</td>
<td>FDCAN Rx FIFO 1 configuration register (FDCAN_RXF1C)</td>
<td></td>
</tr>
<tr>
<td>61.5.31</td>
<td>FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)</td>
<td></td>
</tr>
<tr>
<td>61.5.32</td>
<td>FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)</td>
<td></td>
</tr>
<tr>
<td>61.5.33</td>
<td>FDCAN Rx buffer element size configuration register (FDCAN_RXESC)</td>
<td></td>
</tr>
<tr>
<td>61.5.34</td>
<td>FDCAN Tx buffer configuration register (FDCAN_TXBC)</td>
<td></td>
</tr>
<tr>
<td>61.5.35</td>
<td>FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)</td>
<td></td>
</tr>
<tr>
<td>61.5.36</td>
<td>FDCAN Tx buffer element size configuration register (FDCAN_TXESC)</td>
<td></td>
</tr>
<tr>
<td>61.5.37</td>
<td>FDCAN Tx buffer request pending register (FDCAN_TXBRP)</td>
<td></td>
</tr>
<tr>
<td>61.5.38</td>
<td>FDCAN Tx buffer add request register (FDCAN_TXBAR)</td>
<td></td>
</tr>
<tr>
<td>61.5.39</td>
<td>FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)</td>
<td></td>
</tr>
<tr>
<td>61.5.40</td>
<td>FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)</td>
<td></td>
</tr>
<tr>
<td>61.5.41</td>
<td>FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)</td>
<td></td>
</tr>
<tr>
<td>61.5.42</td>
<td>FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBIE)</td>
<td></td>
</tr>
<tr>
<td>61.5.43</td>
<td>FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE)</td>
<td></td>
</tr>
<tr>
<td>61.5.44</td>
<td>FDCAN Tx event FIFO configuration register (FDCAN_TXEFC)</td>
<td></td>
</tr>
<tr>
<td>61.5.45</td>
<td>FDCAN Tx event FIFO status register (FDCAN_TXEFS)</td>
<td></td>
</tr>
<tr>
<td>61.5.46</td>
<td>FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)</td>
<td></td>
</tr>
<tr>
<td>61.5.47</td>
<td>FDCAN TT trigger memory configuration register (FDCAN_TTTMC)</td>
<td></td>
</tr>
<tr>
<td>61.5.48</td>
<td>FDCAN TT reference message configuration register (FDCAN_TTRMC)</td>
<td></td>
</tr>
<tr>
<td>61.5.49</td>
<td>FDCAN TT operation configuration register (FDCAN_TTOCF)</td>
<td></td>
</tr>
<tr>
<td>61.5.50</td>
<td>FDCAN TT matrix limits register (FDCAN_TTMLM)</td>
<td></td>
</tr>
<tr>
<td>61.5.51</td>
<td>FDCAN TUR configuration register (FDCAN_TURCF)</td>
<td></td>
</tr>
<tr>
<td>61.5.52</td>
<td>FDCAN TT operation control register (FDCAN_TTOCN)</td>
<td></td>
</tr>
<tr>
<td>61.5.53</td>
<td>FDCAN TT global time preset register (FDCAN_TTGTP)</td>
<td></td>
</tr>
<tr>
<td>61.5.54</td>
<td>FDCAN TT time mark register (FDCAN_TTMK)</td>
<td></td>
</tr>
<tr>
<td>61.5.55</td>
<td>FDCAN TT interrupt register (FDCAN_TTIR)</td>
<td></td>
</tr>
<tr>
<td>61.5.56</td>
<td>FDCAN TT interrupt enable register (FDCAN_TTIE)</td>
<td></td>
</tr>
<tr>
<td>61.5.57</td>
<td>FDCAN TT interrupt line select register (FDCAN_TTILS)</td>
<td></td>
</tr>
<tr>
<td>61.5.58</td>
<td>FDCAN TT operation status register (FDCAN_TTOST)</td>
<td></td>
</tr>
</tbody>
</table>
61.5.59 FDCAN TUR numerator actual register (FDCAN_TURNA) ........................................... 2514
61.5.60 FDCAN TT local and global time register (FDCAN_TTLGT) ........................................... 2515
61.5.61 FDCAN TT cycle time and count register (FDCAN_TTCTC) ........................................... 2515
61.5.62 FDCAN TT capture time register (FDCAN_TTCPT) ....................................................... 2516
61.5.63 FDCAN TT cycle sync mark register (FDCAN_TTCSM) .................................................. 2516
61.5.64 FDCAN TT trigger select register (FDCAN_TTTS) ....................................................... 2517
61.5.65 FDCAN register map and reset value table ................................................................. 2518

61.6 CCU registers ................................................. 2523
61.6.1 Clock calibration unit core release register (FDCAN_CCU_CREL) .................................... 2523
61.6.2 Calibration configuration register (FDCAN_CCU_CCFG) .............................................. 2523
61.6.3 Calibration status register (FDCAN_CCU_CSTAT) ....................................................... 2525
61.6.4 Calibration watchdog register (FDCAN_CCU_CWD) ..................................................... 2525
61.6.5 Clock calibration unit interrupt register (FDCAN_CCU_IR) ............................................ 2526
61.6.6 Clock calibration unit interrupt enable register (FDCAN_CCU_IE) ................................ 2527
61.6.7 CCU register map and reset value table ................................................................. 2528

62 USB on-the-go high-speed (OTG_HS) ................................................................. 2529
62.1 Introduction .................................................... 2529
62.2 OTG_HS main features ............................................ 2530
62.2.1 General features ........................................... 2530
62.2.2 Host-mode features ......................................... 2531
62.2.3 Peripheral-mode features .................................... 2531
62.3 OTG_HS implementation ........................................... 2532
62.4 OTG_HS functional description ........................................ 2532
62.4.1 OTG_HS block diagram ......................................... 2532
62.4.2 OTG_HS pin and internal signals ................................... 2533
62.4.3 OTG_HS core .................................................. 2533
62.4.4 Embedded full speed OTG PHY ..................................... 2534
62.4.5 High-speed OTG PHY ........................................... 2534
62.5 OTG_HS dual role device (DRD) ........................................ 2534
62.5.1 ID line detection ............................................. 2534
62.5.2 HNP dual role device ........................................ 2534
62.5.3 SRP dual role device .......................................... 2535
62.6 OTG_HS as a USB peripheral ........................................ 2535
62.6.1 SRP-capable peripheral ........................................ 2535
62.6.2 Peripheral states ............................................. 2536
62.6.3 Peripheral endpoints ........................................... 2537
62.7 OTG_HS as a USB host ........................................... 2539
   62.7.1 SRP-capable host ........................................ 2539
   62.7.2 USB host states .......................................... 2540
   62.7.3 Host channels .......................................... 2541
   62.7.4 Host scheduler ......................................... 2542
62.8 OTG_HS SOF trigger ........................................... 2543
   62.8.1 Host SOFs ............................................. 2544
   62.8.2 Peripheral SOFs ........................................ 2544
62.9 OTG_HS low-power modes .................................... 2544
62.10 OTG_HS Dynamic update of the OTG_HFIR register ........ 2545
62.11 OTG_HS data FIFOs ........................................... 2546
   62.11.1 Peripheral FIFO architecture ......................... 2547
   62.11.2 Host FIFO architecture ................................ 2548
   62.11.3 FIFO RAM allocation ................................... 2549
62.12 OTG_HS interrupts ............................................ 2551
62.13 OTG_HS control and status registers ..................... 2553
   62.13.1 CSR memory map ....................................... 2553
62.14 OTG_HS registers ............................................. 2558
   62.14.1 OTG control and status register (OTG_GOTGCTL) .... 2558
   62.14.2 OTG interrupt register (OTG_GOTGINT) ............... 2561
   62.14.3 OTG AHB configuration register (OTG_GAHBCFG) ...... 2563
   62.14.4 OTG USB configuration register (OTG_GUSBCFG) ...... 2564
   62.14.5 OTG reset register (OTG_GRSTCTL) .................... 2568
   62.14.6 OTG core interrupt register (OTG_GINTSTS) ......... 2570
   62.14.7 OTG interrupt mask register (OTG_GINTMSK) ......... 2574
   62.14.8 OTG receive status debug read register (OTG_GRXSTSR) ... 2578
   62.14.9 OTG receive status debug read [alternate] (OTG_GRXSTSR) ... 2579
   62.14.10 OTG status read and pop registers (OTG_GRXSTSP) ... 2580
   62.14.11 OTG status read and pop registers [alternate] (OTG_GRXSTSP) ... 2581
   62.14.12 OTG receive FIFO size register (OTG_GRXFSIZ) .... 2582
   62.14.13 OTG host non-periodic transmit FIFO size register (OTG_HNPTXFSIZ)/Endpoint 0 Transmit FIFO size (OTG_DIEPTXF0) ................. 2582
   62.14.14 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS) ........................................... 2583
   62.14.15 OTG general core configuration register (OTG_GCCFG) .... 2584
62.14.16 OTG core ID register (OTG_CID) ........................................... 2586
62.14.17 OTG core LPM configuration register (OTG_GLPMCFG) .............. 2586
62.14.18 OTG host periodic transmit FIFO size register
(OTG_HPTXFSIZ) ............................................................. 2590
62.14.19 OTG device IN endpoint transmit FIFO x size register
(OTG_DIEPTXFx) ............................................................ 2590
62.14.20 Host-mode registers .......................................................... 2591
62.14.21 OTG host configuration register (OTG_HCFG) ......................... 2591
62.14.22 OTG host frame interval register (OTG_HFIR) ......................... 2592
62.14.23 OTG host frame number/frame time remaining register
(OTG_HFNUM) ................................................................. 2593
62.14.24 OTG_Host periodic transmit FIFO/queue status register
(OTG_HPTXSTS) ............................................................... 2594
62.14.25 OTG host all channels interrupt register (OTG_HAINT) ............. 2595
62.14.26 OTG host all channels interrupt mask register
(OTG_HAINTMSK) ............................................................ 2595
62.14.27 OTG host frame list base address register
(OTG_HFLBADDR) .............................................................. 2596
62.14.28 OTG host port control and status register (OTG_HPRT) ............. 2596
62.14.29 OTG host channel x characteristics register (OTG_HCCHARx) ...... 2599
62.14.30 OTG host channel x split control register (OTG_HCSPLTx) ........ 2600
62.14.31 OTG host channel x interrupt register (OTG_HCINTx) ............... 2601
62.14.32 OTG host channel x interrupt mask register (OTG_HCINTMSKx) .. 2602
62.14.33 OTG host channel x transfer size register (OTG_HCTSIZx) ......... 2604
62.14.34 OTG host channel x transfer size register (OTG_HCTSIZSGx) .... 2605
62.14.35 OTG host channel x DMA address register in buffer DMA [alternate]
(OTG_HCDMAx) ............................................................... 2607
62.14.36 OTG host channel x DMA address register in scatter/gather DMA
[alternate] (OTG_HCDMASGx) .................................................. 2607
62.14.37 OTG host channel-n DMA address buffer register
(OTG_HCDMABx) .............................................................. 2608
62.14.38 Device-mode registers ..................................................... 2609
62.14.39 OTG device configuration register (OTG_DCFG) ....................... 2609
62.14.40 OTG device control register (OTG_DCTL) ............................... 2611
62.14.41 OTG device status register (OTG_DSTS) .................................. 2613
62.14.42 OTG device IN endpoint common interrupt mask register
(OTG_DIEPMSK) .............................................................. 2614
62.14.43 OTG device OUT endpoint common interrupt mask register
(OTG_DOEPMASK) ............................................................ 2615
62.14.44 OTG device all endpoints interrupt register (OTG_DAINT) ......... 2616
62.14.45 OTG all endpoints interrupt mask register (OTG_DAINTMSK) ........................................... 2617
62.14.46 OTG device VBUS discharge time register (OTG_DVBUSDIS) .................................................. 2618
62.14.47 OTG device VBUS pulsing time register (OTG_DVBUSPULSE) .................................................... 2618
62.14.48 OTG device threshold control register (OTG_DTHRCTL) ......................................................... 2619
62.14.49 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK) .................... 2620
62.14.50 OTG device each endpoint interrupt register (OTG_DEACHINT) .............................................. 2620
62.14.51 OTG device each endpoint interrupt mask register (OTG_DEACHINTMSK) .................................. 2621
62.14.52 OTG device each IN endpoint-1 interrupt mask register (OTG_HS_DIEPEACHMSK1) ................... 2621
62.14.53 OTG device each OUT endpoint-1 interrupt mask register (OTG_HS_DOEPEACHMSK1) ................ 2622
62.14.54 OTG device IN endpoint x control register (OTG_DIEPCTLx) .................................................. 2624
62.14.55 OTG device IN endpoint x interrupt register (OTG_DIEPINTx) .................................................. 2626
62.14.56 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0) .......................................... 2628
62.14.57 OTG device IN endpoint x DMA address register (OTG_DIEPDMAx) .......................................... 2628
62.14.58 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx) .................................. 2629
62.14.59 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx) .......................................... 2629
62.14.60 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0) .................................... 2630
62.14.61 OTG device OUT endpoint x interrupt register (OTG_DOEPIIINTx) .......................................... 2632
62.14.62 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0) ....................................... 2634
62.14.63 OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx) ...................................... 2635
62.14.64 OTG device OUT endpoint x control register (OTG_DOEPCTLx) .............................................. 2635
62.14.65 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx) ....................................... 2638
62.14.66 OTG power and clock gating control register (OTG_PCGCCTL) .............................................. 2639
62.14.67 OTG_HS register map ........................................... 2640
62.15 OTG_HS programming model ........................................... 2652
  62.15.1 Core initialization ........................................... 2652
  62.15.2 Host initialization ........................................... 2652
63 HDMI-CEC controller (CEC) ........................................ 2714
   63.1 Introduction ............................................... 2714
   63.2 HDMI-CEC controller main features ....................... 2714
   63.3 HDMI-CEC functional description ......................... 2715
      63.3.1 HDMI-CEC pin and internal signals .................... 2715
      63.3.2 HDMI-CEC block diagram ............................... 2716
      63.3.3 Message description .................................. 2716
      63.3.4 Bit timing ............................................ 2717
   63.4 Arbitration ................................................ 2717
      63.4.1 SFT option bit ....................................... 2719
   63.5 Error handling ............................................ 2719
      63.5.1 Bit error ............................................. 2719
      63.5.2 Message error ........................................ 2720
      63.5.3 Bit rising error (BRE) ................................ 2720
      63.5.4 Short bit period error (SBPE) ......................... 2720
      63.5.5 Long bit period error (LBPE) ......................... 2720
      63.5.6 Transmission error detection (TXERR) ................ 2722
   63.6 HDMI-CEC interrupts ....................................... 2723
   63.7 HDMI-CEC registers ....................................... 2724
      63.7.1 CEC control register (CEC_CR) ....................... 2724
      63.7.2 CEC configuration register (CEC_CFGR) ........…… 2725
      63.7.3 CEC Tx data register (CEC_TXDR) .................... 2727
      63.7.4 CEC Rx data register (CEC_RXDR) .................... 2727
      63.7.5 CEC interrupt and status register (CEC_ISR) ....... 2727
      63.7.6 CEC interrupt enable register (CEC_IER) .......... 2729
      63.7.7 HDMI-CEC register map ................................ 2731

64 Debug infrastructure .......................................... 2732
   64.1 Introduction ............................................... 2732
64.2 Debug infrastructure features ............................................. 2733
64.3 Debug infrastructure functional description ......................... 2734
   64.3.1 Debug infrastructure block diagram ............................... 2734
   64.3.2 Debug infrastructure pins and internal signals .................. 2734
   64.3.3 Debug infrastructure powering, clocking and reset .............. 2736
64.4 Debug access port functional description ............................. 2738
   64.4.1 Serial-wire and JTAG debug port (SWJ-DP) ....................... 2738
   64.4.2 Access ports .................................................. 2752
64.5 Trace and debug subsystem functional description .................. 2757
   64.5.1 System ROM tables ............................................. 2757
   64.5.2 Cross trigger interfaces (CTI) and matrix (CTM) ................. 2765
   64.5.3 Trace funnel (CSTF) ........................................... 2785
   64.5.4 Embedded trace FIFO (ETF) .................................... 2796
   64.5.5 Trace port interface unit (TPIU) ................................ 2819
   64.5.6 Serial-wire output (SWO) ...................................... 2838
   64.5.7 Microcontroller debug unit (DBGMCU) ............................ 2851
64.6 Cortex-M7 debug functional description ............................... 2857
   64.6.1 Cortex-M7 ROM tables ......................................... 2858
   64.6.2 Cortex-M7 data watchpoint and trace unit (DWT) ................. 2872
   64.6.3 Cortex-M7 instrumentation trace macrocell (ITM) ................. 2886
   64.6.4 Cortex-M7 breakpoint unit (FPB) ................................ 2895
   64.6.5 Cortex-M7 embedded trace macrocell (ETM) ....................... 2903
   64.6.6 Cortex-M7 cross trigger interface (CTI) ........................ 2935
64.7 References for debug infrastructure .................................. 2935

65 Device electronic signature ................................................. 2936
   65.1 Unique device ID register (96 bits) ................................ 2936
   65.2 Flash size ....................................................... 2937
   65.3 Package data register ............................................. 2937

66 Revision history .......................................................... 2939
List of tables

Table 1. Availability of security features ................................................................. 98
Table 2. Bus-master-to-bus-slave interconnect ......................................................... 99
Table 3. ASIB configuration ....................................................................................... 107
Table 4. AMIB configuration ....................................................................................... 107
Table 5. AXI interconnect register map and reset values ........................................... 117
Table 6. Register boundary addresses ....................................................................... 129
Table 7. Boot modes .................................................................................................. 135
Table 8. RAMECC internal input/output signals ....................................................... 139
Table 9. ECC controller mapping ............................................................................... 139
Table 10. RAMECC register map and reset values ................................................... 144
Table 11. FLASH internal input/output signals .......................................................... 147
Table 12. Flash memory organization (STM32H7A3xl/7B3xl devices) ..................... 150
Table 13. Flash memory organization (STM32H7B0 devices) ................................... 151
Table 14. Flash memory organization (STM32H7A3xG devices) .............................. 151
Table 15. FLASH recommended number of wait states and programming delay ......... 158
Table 16. Flash memory OTP organization ................................................................. 169
Table 17. Read-only public data organization ............................................................. 171
Table 18. FLASH AXI interface memory map and the swapping option .................... 171
Table 19. Flash register map vs swapping option ...................................................... 174
Table 20. Option byte organization ............................................................................. 179
Table 21. Flash interface register protection summary .............................................. 185
Table 22. RDP value vs readout protection level ...................................................... 187
Table 23. Protection vs RDP Level ............................................................................. 188
Table 24. RDP transition and its effects .................................................................... 190
Table 25. Effect of low-power modes on the embedded Flash memory ...................... 194
Table 26. Flash interrupt request ................................................................................ 201
Table 27. Register map and reset value table ............................................................. 244
Table 28. List of preferred terms ................................................................................ 250
Table 29. Summary of Flash protected areas access rights ....................................... 256
Table 30. PWR input/output signals connected to package pins or balls .................... 258
Table 31. PWR internal input/output signals ............................................................... 259
Table 32. Supply configuration control ...................................................................... 264
Table 33. Low-power mode summary ....................................................................... 283
Table 34. PDDS_SRD and RETDS_CD low-power mode control ............................... 286
Table 35. Low-power exit mode flags ....................................................................... 287
Table 36. CSleep mode .............................................................................................. 297
Table 37. CStop mode ............................................................................................... 298
Table 38. DStop mode overview ................................................................................. 298
Table 39. DStop or DStop2 mode ............................................................................... 299
Table 40. Stop mode operation .................................................................................. 301
Table 41. Memory shut-off block selection ............................................................... 301
Table 42. Stop mode .................................................................................................. 302
Table 43. Standby and Stop flags .............................................................................. 304
Table 44. Standby mode ............................................................................................ 304
Table 45. Overview of low-power mode monitoring pins ......................................... 304
Table 46. GPIO state according to CPU and domain state ........................................ 305
Table 47. Power control register map and reset values ............................................. 316
Table 48. BDMA2 and DMAMUX2 initialization sequence (DMAMUX2_INIT) ........ 325
Table 49. LPUART1 Initial programming (LPUART1_INIT) .................................................. 327
Table 50. LPUART1 start programming (LPUART1_Start) ................................................ 327
Table 51. RCC input/output signals connected to package pins or balls ............................. 331
Table 52. RCC internal input/output signals ................................................................. 332
Table 53. Reset distribution summary ................................................................. 335
Table 54. Reset source identification (RCC_RSR) ...................................................... 336
Table 55. Ratio between clock timer and pclk .......................................................... 355
Table 56. STOPWUCK and STOPKERWUCK description ........................................... 356
Table 57. HSikerON and CSikerON behavior ........................................................... 357
Table 58. Kernel clock distribution overview .......................................................... 358
Table 59. System states overview .............................................................................. 372
Table 60. Peripheral clock enabling for the CPU domain peripherals ......................... 375
Table 61. Peripheral clock enabling for SmartRun domain peripherals ....................... 377
Table 62. Interrupt sources and control ..................................................................... 381
Table 63. RCC register map and reset values ............................................................ 470
Table 64. CRS features ......................................................................................... 477
Table 65. CRS internal input/output signals .............................................................. 478
Table 66. Effect of low-power modes on CRS .......................................................... 482
Table 67. Interrupt control bits .............................................................................. 482
Table 68. CRS register map and reset values ............................................................ 488
Table 69. HSEM internal input/output signals ........................................................... 490
Table 70. Authorized AHB bus master ID .................................................. 495
Table 71. HSEM register map and reset values ....................................................... 501
Table 72. Port bit configuration table .................................................................... 504
Table 73. GPIO register map and reset values ......................................................... 519
Table 74. SYSCFG register map and reset values ..................................................... 533
Table 75. Peripherals interconnect matrix (CD domain) ............................................ 535
Table 76. Peripherals interconnect matrix (SRD domain) ......................................... 536
Table 77. Peripherals interconnect matrix details .................................................... 536
Table 78. EXTI wakeup inputs ................................................................................ 548
Table 79. EXTI pending requests clear inputs .......................................................... 551
Table 80. MDMA .................................................................................................... 553
Table 81. DMAMUX1, DMA1, DMA2 and BDMA1 connections .............................. 555
Table 82. DMAMUX2 and BDMA2 connections ...................................................... 560
Table 83. DMA internal input/output signals ............................................................ 565
Table 84. DMA interrupt requests ........................................................................... 572
Table 85. DMA register map and reset values .......................................................... 590
Table 86. DMA internal input/output signals ............................................................ 594
Table 87. Source and destination address ............................................................... 596
Table 88. Source and destination address registers in double-buffer mode (DBM = 1) .... 602
Table 89. Packing/unpacking and endian behavior (bit PINC = MINC = 1) ............... 603
Table 90. Restriction on NDT versus PSIZE and MSIZE ........................................ 603
Table 91. FIFO threshold configurations ............................................................... 606
Table 92. Possible DMA configurations ............................................................... 610
Table 93. DMA interrupt requests ........................................................................... 612
Table 94. DMA register map and reset values .......................................................... 623
Table 95. BDMA1 and BDMA2 implementation ...................................................... 628
Table 96. BDMA internal input/output signals ......................................................... 631
Table 97. Programmable data width and endian behavior (when PINC = MINC = 1) .... 637
Table 98. BDMA interrupt requests ........................................................................... 639
Table 99. BDMA register map and reset values .......................................................... 649
Table 100. DMAMUX1 and DMAMUX2 instantiation ............................................ 653
Table 101. DMAMUX1: assignment of multiplexer inputs to resources ........................................ 654
Table 102. DMAMUX1: assignment of trigger inputs to resources ........................................... 655
Table 103. DMAMUX1: assignment of synchronization inputs to resources ............................... 655
Table 104. DMAMUX2: assignment of multiplexer inputs to resources .................................... 655
Table 105. DMAMUX2: assignment of trigger inputs to resources ............................................ 657
Table 106. DMAMUX2: assignment of synchronization inputs to resources ............................. 657
Table 107. DMAMUX signals ........................................................................................................ 659
Table 108. DMAMUX interrupts .................................................................................................. 663
Table 109. DMAMUX register map and reset values ................................................................. 672
Table 110. DMA2D internal input/output signals ....................................................................... 676
Table 111. Supported color mode in input .................................................................................... 677
Table 112. Data order in memory ............................................................................................... 679
Table 113. Alpha mode configuration .......................................................................................... 679
Table 114. Supported CLUT color mode ....................................................................................... 680
Table 115. CLUT data order in memory ....................................................................................... 680
Table 116. Supported color mode in output .................................................................................. 681
Table 117. Data order in memory ............................................................................................... 682
Table 118. Standard data order in memory .................................................................................. 683
Table 119. Output FIFO byte reordering steps ......................................................................... 684
Table 120. MCU order in memory ............................................................................................. 689
Table 121. DMA2D interrupt requests ......................................................................................... 690
Table 122. DMA2D register map and reset values ..................................................................... 710
Table 123. NVIC ........................................................................................................................ 713
Table 124. EXTI event input configurations and register control ............................................. 725
Table 125. Configurable event input asynchronous edge detector reset .................................... 730
Table 126. EXTI Event input mapping ......................................................................................... 732
Table 127. Masking functionality ............................................................................................... 732
Table 128. Asynchronous interrupt/event controller register map and reset values .................. 752
Table 129. Graphic MMU interrupt requests ............................................................................... 764
Table 130. Graphic MMU register map and reset values ........................................................... 772
Table 131. CRC internal input/output signals ............................................................................. 774
Table 132. CRC register map and reset values ........................................................................... 778
Table 133. FMC pins .................................................................................................................. 782
Table 134. FMC bank mapping options ...................................................................................... 785
Table 135. NOR/PSRAM bank selection ..................................................................................... 786
Table 136. NOR/PSRAM External memory address ................................................................. 786
Table 137. NAND memory mapping and timing registers ......................................................... 786
Table 138. NAND bank selection .............................................................................................. 786
Table 139. SDRAM bank selection ............................................................................................ 787
Table 140. SDRAM bank selection ............................................................................................ 787
Table 141. SDRAM address mapping ........................................................................................ 787
Table 142. SDRAM address mapping with 8-bit data bus width .............................................. 788
Table 143. SDRAM address mapping with 16-bit data bus width ............................................ 789
Table 144. SDRAM address mapping with 32-bit data bus width ............................................ 790
Table 145. Programmable NOR/PSRAM access parameters ................................................ 791
Table 146. Non-multiplexed I/O NOR Flash memory .............................................................. 792
Table 147. Non-multiplexed I/Os PSRAM/SRAM ..................................................................... 793
Table 148. 16-bit multiplexed I/O PSRAM ................................................................................. 793
Table 149. NOR Flash/PSRAM: Example of supported memories and transactions .............. 794
Table 150. FMC_BCRx bitfields (mode 1) ................................................................................... 797
Table 151. FMC_BTRx bitfields (mode 1) ................................................................................... 798
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>152</td>
<td>FMC_BCRx bitfields (mode A)</td>
<td>800</td>
</tr>
<tr>
<td>153</td>
<td>FMC_BTRx bitfields (mode A)</td>
<td>801</td>
</tr>
<tr>
<td>154</td>
<td>FMC_BWTRx bitfields (mode A)</td>
<td>801</td>
</tr>
<tr>
<td>155</td>
<td>FMC_BCRx bitfields (mode 2/B)</td>
<td>803</td>
</tr>
<tr>
<td>156</td>
<td>FMC_BTRx bitfields (mode 2/B)</td>
<td>804</td>
</tr>
<tr>
<td>157</td>
<td>FMC_BWTRx bitfields (mode 2/B)</td>
<td>804</td>
</tr>
<tr>
<td>158</td>
<td>FMC_BCRx bitfields (mode C)</td>
<td>806</td>
</tr>
<tr>
<td>159</td>
<td>FMC_BTRx bitfields (mode C)</td>
<td>807</td>
</tr>
<tr>
<td>160</td>
<td>FMC_BWTRx bitfields (mode C)</td>
<td>807</td>
</tr>
<tr>
<td>161</td>
<td>FMC_BCRx bitfields (mode D)</td>
<td>809</td>
</tr>
<tr>
<td>162</td>
<td>FMC_BTRx bitfields (mode D)</td>
<td>809</td>
</tr>
<tr>
<td>163</td>
<td>FMC_BWTRx bitfields (mode D)</td>
<td>810</td>
</tr>
<tr>
<td>164</td>
<td>FMC_BCRx bitfields (Muxed mode)</td>
<td>812</td>
</tr>
<tr>
<td>165</td>
<td>FMC_BTRx bitfields (Muxed mode)</td>
<td>812</td>
</tr>
<tr>
<td>166</td>
<td>FMC_BCRx bitfields (Synchronous multiplexed read mode)</td>
<td>818</td>
</tr>
<tr>
<td>167</td>
<td>FMC_BTRx bitfields (Synchronous multiplexed read mode)</td>
<td>818</td>
</tr>
<tr>
<td>168</td>
<td>FMC_BCRx bitfields (Synchronous multiplexed write mode)</td>
<td>819</td>
</tr>
<tr>
<td>169</td>
<td>FMC_BTRx bitfields (Synchronous multiplexed write mode)</td>
<td>820</td>
</tr>
<tr>
<td>170</td>
<td>Programmable NAND Flash access parameters</td>
<td>830</td>
</tr>
<tr>
<td>171</td>
<td>8-bit NAND Flash memory</td>
<td>830</td>
</tr>
<tr>
<td>172</td>
<td>16-bit NAND Flash memory</td>
<td>831</td>
</tr>
<tr>
<td>173</td>
<td>Supported memories and transactions</td>
<td>831</td>
</tr>
<tr>
<td>174</td>
<td>ECC result relevant bits</td>
<td>841</td>
</tr>
<tr>
<td>175</td>
<td>SDRAM signals</td>
<td>842</td>
</tr>
<tr>
<td>176</td>
<td>FMC register map</td>
<td>859</td>
</tr>
<tr>
<td>177</td>
<td>Command/address phase description</td>
<td>872</td>
</tr>
<tr>
<td>178</td>
<td>OCTOSPI interrupt requests</td>
<td>889</td>
</tr>
<tr>
<td>179</td>
<td>OCTOSPI register map and reset values</td>
<td>913</td>
</tr>
<tr>
<td>180</td>
<td>OCTOSPI register map and reset values</td>
<td>923</td>
</tr>
<tr>
<td>181</td>
<td>DLYB internal input/output signals</td>
<td>925</td>
</tr>
<tr>
<td>182</td>
<td>Delay block control</td>
<td>925</td>
</tr>
<tr>
<td>183</td>
<td>DLYB register map and reset values</td>
<td>928</td>
</tr>
<tr>
<td>184</td>
<td>ADC features</td>
<td>931</td>
</tr>
<tr>
<td>185</td>
<td>ADC input/output pins</td>
<td>933</td>
</tr>
<tr>
<td>186</td>
<td>ADC internal input/output signals</td>
<td>933</td>
</tr>
<tr>
<td>187</td>
<td>ADC interconnection</td>
<td>934</td>
</tr>
<tr>
<td>188</td>
<td>Configuring the trigger polarity for regular external triggers</td>
<td>953</td>
</tr>
<tr>
<td>189</td>
<td>Configuring the trigger polarity for injected external triggers</td>
<td>953</td>
</tr>
<tr>
<td>190</td>
<td>ADC1 and ADC2 - External triggers for regular channels</td>
<td>954</td>
</tr>
<tr>
<td>191</td>
<td>ADC1 and ADC2 - External triggers for injected channels</td>
<td>955</td>
</tr>
<tr>
<td>192</td>
<td>TSAR timings depending on resolution</td>
<td>967</td>
</tr>
<tr>
<td>193</td>
<td>Offset computation versus data resolution</td>
<td>970</td>
</tr>
<tr>
<td>194</td>
<td>16-bit data formats</td>
<td>973</td>
</tr>
<tr>
<td>195</td>
<td>Numerical examples for 16-bit format (bold indicates saturation)</td>
<td>973</td>
</tr>
<tr>
<td>196</td>
<td>Analog watchdog channel selection</td>
<td>981</td>
</tr>
<tr>
<td>197</td>
<td>Analog watchdog 1,2,3 comparison</td>
<td>982</td>
</tr>
<tr>
<td>198</td>
<td>Oversampler operating modes summary</td>
<td>990</td>
</tr>
<tr>
<td>199</td>
<td>ADC interrupts per each ADC</td>
<td>1009</td>
</tr>
<tr>
<td>200</td>
<td>DELAY bits versus ADC resolution</td>
<td>1048</td>
</tr>
<tr>
<td>201</td>
<td>ADC global register map</td>
<td>1050</td>
</tr>
<tr>
<td>202</td>
<td>ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC)</td>
<td>1050</td>
</tr>
</tbody>
</table>
List of tables

Table 203. ADC register map and reset values (master and slave ADC common registers) offset =0x300) ........................................... 1053
Table 204. DTS internal input/output signals ........................................... 1055
Table 205. Sampling time configuration ........................................... 1058
Table 206. Trigger configuration ........................................... 1059
Table 207. Temperature sensor behavior in low-power modes .......................... 1061
Table 208. Interrupt control bits ........................................... 1062
Table 209. DTS register map and reset values ........................................... 1070
Table 210. DAC implementation ........................................... 1072
Table 211. DAC input/output pins ........................................... 1074
Table 212. DAC internal input/output signals ........................................... 1074
Table 213. DAC1 interconnection ........................................... 1075
Table 214. DAC2 interconnection ........................................... 1075
Table 215. Sample and refresh timings ........................................... 1082
Table 216. Channel output modes summary ........................................... 1083
Table 217. Effect of low-power modes on DAC1 ........................................... 1089
Table 218. Effect of low-power modes on DAC2 ........................................... 1090
Table 219. DAC interrupts ........................................... 1090
Table 220. DAC register map and reset values ........................................... 1107
Table 221. VREF buffer modes ........................................... 1109
Table 222. VREFBUF register map and reset values ........................................... 1111
Table 223. COMP input/output internal signals ........................................... 1114
Table 224. COMP input/output pins ........................................... 1115
Table 225. COMP interconnection ........................................... 1115
Table 226. COMP1_OUT assignment to GPIOs ........................................... 1117
Table 227. COMP2_OUT assignment to GPIOs ........................................... 1118
Table 228. Comparator behavior in the low-power modes ........................................... 1120
Table 229. Interrupt control bits ........................................... 1121
Table 230. COMP register map and reset values ........................................... 1129
Table 231. COMP interconnection ........................................... 1131
Table 232. Operating modes and calibration ........................................... 1139
Table 233. Effect of low-power modes on the OPAMP ........................................... 1141
Table 234. OPAMP register map and reset values ........................................... 1148
Table 235. DFSDMx implementation ........................................... 1151
Table 236. DFSDM1 external pins ........................................... 1153
Table 237. DFSDM1 internal signals ........................................... 1153
Table 238. DFSDM2 external pins ........................................... 1153
Table 239. Effect of low-power modes on DAC1 ........................................... 1154
Table 240. DFSDM2 internal signals ........................................... 1154
Table 241. DFSDM triggers connection ........................................... 1154
Table 242. DFSDM1 break connection ........................................... 1154
Table 243. Filter maximum output resolution (peak data values from filter output) for some IOSR values ........................................... 1169
Table 244. Integrator maximum output resolution (peak data values from integrator output) for some IOSR values and FOSR = 256 and Sinc3 filter type (largest data) ........................................... 1170
Table 245. DFSDM interrupt requests ........................................... 1178
Table 246. DFSDM register map and reset values ........................................... 1199
Table 247. DCMI input/output pins ........................................... 1215
Table 248. DCMI internal input/output signals ........................................... 1215
Table 249. Positioning of captured data bytes in 32-bit words (8-bit width) ........................................... 1217
Table 250. Positioning of captured data bytes in 32-bit words (10-bit width) ........................................... 1217
Table 251. Positioning of captured data bytes in 32-bit words (12-bit width) ........................................... 1217
Table 252. Positioning of captured data bytes in 32-bit words (14-bit width) ........................................ 1218
Table 253. Data storage in monochrome progressive video format ..................................................... 1223
Table 254. Data storage in RGB progressive video format ............................................................... 1224
Table 255. Data storage in YCbCr progressive video format ............................................................... 1224
Table 256. Data storage in YCbCr progressive video format - Y extraction mode .............................. 1225
Table 257. DCMI interrupts ............................................................................................................. 1226
Table 258. DCMI register map and reset values ................................................................................. 1237
Table 259. PSSI input/output pins .................................................................................................... 1240
Table 260. PSSI internal input/output signals ..................................................................................... 1240
Table 261. Positioning of captured data bytes in 32-bit words (8-bit width) ..................................... 1241
Table 262. Positioning of captured data bytes in 32-bit words (16-bit width) ................................... 1242
Table 263. PSSI interrupt requests .................................................................................................... 1245
Table 264. PSSI register map and reset values ................................................................................... 1252
Table 265. LCD-TFT internal signals ................................................................................................. 1254
Table 266. LTDC pins and signal interface ....................................................................................... 1255
Table 267. Clock domain for each register ....................................................................................... 1256
Table 268. LTDC register access and update durations ....................................................................... 1257
Table 269. Pixel data mapping versus color format ........................................................................... 1261
Table 270. LTDC interrupt requests .................................................................................................. 1265
Table 271. LTDC register map and reset values ................................................................................ 1285
Table 272. JPEG internal signals ..................................................................................................... 1289
Table 273. JPEG codec interrupt requests ......................................................................................... 1293
Table 274. JPEG codec register map and reset values ..................................................................... 1306
Table 275. RNG internal input/output signals ................................................................................... 1309
Table 276. RNG interrupt requests .................................................................................................. 1317
Table 277. RNG configurations ....................................................................................................... 1318
Table 278. RNG register map and reset map ................................................................................... 1323
Table 279. CRYP internal input/output signals ............................................................................... 1327
Table 280. Counter mode initialization vector ................................................................................... 1352
Table 281. GCM last block definition ............................................................................................... 1355
Table 282. GCM mode IV registers initialization ............................................................................... 1355
Table 283. CCM mode IV registers initialization ............................................................................... 1362
Table 284. DES/TDES data swapping feature ................................................................................... 1368
Table 285. AES data swapping feature ............................................................................................. 1370
Table 286. Key registers CRYP_KxR/LR endianness (TDES K1/2/3 and AES 128/192/256-bit keys) ........................................................................................................... 1370
Table 287. Initialization vector registers CRYP_IVxR endianness ....................................................... 1371
Table 288. Cryptographic processor configuration for memory-to-peripheral DMA transfers ........ 1372
Table 289. Cryptographic processor configuration for peripheral-to-memory DMA transfers ....... 1373
Table 290. CRYP interrupt requests .................................................................................................. 1374
Table 291. Processing latency for ECB, CBC and CTR ..................................................................... 1376
Table 292. Processing time (in clock cycle) for GCM and CCM per 128-bit block ......................... 1376
Table 293. CRYP register map and reset values ............................................................................... 1390
Table 294. HASH internal input/output signals ................................................................................ 1395
Table 295. Hash processor outputs .................................................................................................. 1398
Table 296. HASH interrupt requests .................................................................................................. 1405
Table 297. Processing time (in clock cycle) ....................................................................................... 1405
Table 298. HASH1 register map and reset values ............................................................................. 1415
Table 299. OTFDEC internal input/output signals ............................................................................... 1418
Table 300. OTFDEC interrupt requests ............................................................................................. 1423
Table 350. Effect of low-power modes on RTC

Table 348. PI8 configuration

Table 347. RTC pin PC13 configuration

Table 345. RTC internal input/output signals

Table 344. RTC input/output pins

Table 343. IWDG register map and reset values

Table 342. IWDG internal input/output signals

Table 341. WWDG register map and reset values

Table 337. Effect of low-power modes on the LPTIM

Table 339. LPTIM register map and reset values

Table 338. Interrupt events

Table 337. LPTIM register map and reset values

Table 336. Encoder counting scenarios

Table 335. LPTIM3 external trigger connection

Table 334. LPTIM2 input 2 connection

Table 333. LPTIM2 input 1 connection

Table 332. LPTIM1 input 2 connection

Table 331. LPTIM1 input 1 connection

Table 330. LPTIM external trigger connection

Table 329. LPTIM2 external trigger connection

Table 328. LPTIM1 external trigger connection

Table 327. LPTIM internal signals

Table 326. STM32H7A3/7B3 and STM32H7B0 LPTIM features

Table 325. TIM13/TIM14 register map and reset values

Table 324. TIMx register map and reset values

Table 323. TIM16/TIM17 register map and reset values

Table 322. Output control bits for complementary OCx and OCxN channels with break feature (TIM15)

Table 321. TIM15 register map and reset values

Table 320. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)

Table 319. TIMx Internal trigger connection

Table 318. Break protection disarming conditions

Table 317. TIM13/TIM14 register map and reset values

Table 316. Output control bit for standard OCx channels.

Table 315. TIM12 register map and reset values

Table 314. Output control bit for standard OCx channels.

Table 313. TIMx internal trigger connection

Table 312. TIM2/TIM3/TIM4/TIM5 register map and reset values

Table 311. Output control bit for standard OCx channels.

Table 310. TIMx internal trigger connection

Table 309. Counting direction versus encoder signals

Table 308. TIM8 register map and reset values

Table 307. TIM1 register map and reset values

Table 306. Output control bits for complementary OCx and OCxN channels with break feature

Table 305. TIMx internal trigger connection

Table 304. Counting direction versus encoder signals

Table 303. Break protection disarming conditions

Table 302. Behavior of timer outputs versus BRK/BRK2 inputs

Table 301. OTFDEC register map and reset values

Table 300. TIMx register map and reset values

Table 299. STM32H7A3/7B3 and STM32H7B0 LPTIM features

Table 298. Output control bits for complementary OCx and OCxN channels with break feature

Table 297. STM32H7A3/7B3 and STM32H7B0 LPTIM features

Table 296. Output control bits for complementary OCx and OCxN channels with break feature

Table 295. TIM2/TIM3/TIM4/TIM5 register map and reset values

Table 294. Output control bits for complementary OCx and OCxN channels with break feature

Table 293. TIM16/TIM17 register map and reset values

Table 292. Output control bits for complementary OCx and OCxN channels with break feature

Table 291. TIM15 register map and reset values

Table 290. Output control bits for complementary OCx and OCxN channels with break feature

Table 289. Output control bits for complementary OCx and OCxN channels with break feature

Table 288. Output control bits for complementary OCx and OCxN channels with break feature

Table 287. Output control bits for complementary OCx and OCxN channels with break feature

Table 286. Output control bits for complementary OCx and OCxN channels with break feature

Table 285. Output control bits for complementary OCx and OCxN channels with break feature

Table 284. Output control bits for complementary OCx and OCxN channels with break feature

Table 283. Output control bits for complementary OCx and OCxN channels with break feature

Table 282. Output control bits for complementary OCx and OCxN channels with break feature

Table 281. Output control bits for complementary OCx and OCxN channels with break feature

Table 280. Output control bits for complementary OCx and OCxN channels with break feature

Table 279. Output control bits for complementary OCx and OCxN channels with break feature

Table 278. Output control bits for complementary OCx and OCxN channels with break feature

Table 277. Output control bits for complementary OCx and OCxN channels with break feature

Table 276. Output control bits for complementary OCx and OCxN channels with break feature

Table 275. Output control bits for complementary OCx and OCxN channels with break feature

Table 274. Output control bits for complementary OCx and OCxN channels with break feature

Table 273. Output control bits for complementary OCx and OCxN channels with break feature

Table 272. Output control bits for complementary OCx and OCxN channels with break feature

Table 271. Output control bits for complementary OCx and OCxN channels with break feature

Table 270. Output control bits for complementary OCx and OCxN channels with break feature

Table 269. Output control bits for complementary OCx and OCxN channels with break feature

Table 268. Output control bits for complementary OCx and OCxN channels with break feature

Table 267. Output control bits for complementary OCx and OCxN channels with break feature

Table 266. Output control bits for complementary OCx and OCxN channels with break feature

Table 265. Output control bits for complementary OCx and OCxN channels with break feature

Table 264. Output control bits for complementary OCx and OCxN channels with break feature

Table 263. Output control bits for complementary OCx and OCxN channels with break feature

Table 262. Output control bits for complementary OCx and OCxN channels with break feature

Table 261. Output control bits for complementary OCx and OCxN channels with break feature

Table 260. Output control bits for complementary OCx and OCxN channels with break feature

Table 259. Output control bits for complementary OCx and OCxN channels with break feature

Table 258. Output control bits for complementary OCx and OCxN channels with break feature

Table 257. Output control bits for complementary OCx and OCxN channels with break feature
Table 368. I2C-SMBus specification clock timings ................................................. 1904
Table 367. I2C configuration ................................................................. 1893
Table 372. SMBus timeout specifications ......................................................... 1917
Table 370. Examples of timings settings for fI2CCLK = 16 MHz ......................... 1914
Table 369. Examples of timing settings for fI2CCLK = 8 MHz ........................... 1914
Table 366. I2C-SMBus specification data setup and hold times ......................... 1889
Table 374. Examples of TIMEOUTA settings for various i2c_ker_ck frequencies ..... 1920
Table 373. SMBus with PEC configuration .................................................. 1919
Table 365. Comparison of analog vs. digital filters ........................................ 1886
Table 364. I2C internal input/output signals .................................................... 1884
Table 363. I2C input/output pins .................................................................. 1884
Table 362. STM32H7A3/7B3 and STM32H7B0 I2C implementation .................... 1882
Table 361. TAMP register map and reset values .............................................. 1879
Table 360. Interrupt requests ................................................................... 1879
Table 359. TAMP pins functionality over modes .............................................. 1865
Table 358. Effect of low-power modes on TAMP .......................................... 1863
Table 357. Minimum ATPER value .............................................................. 1863
Table 356. TAMP interconnection ................................................................. 1860
Table 355. TAMP internal input/output signals ............................................... 1860
Table 354. TAMP input/output pins ............................................................... 1860
Table 353. RTC register map and reset values .............................................. 1856
Table 352. Interrupt requests ................................................................... 1835
Table 351. RTC pins functionality over modes .............................................. 1835
Table 454. 4-bit mode Start, interrupt, and CRC-status Signaling detection .......................... 2352
Table 455. CMD12 use cases ......................................................... 2356
Table 456. SDMMC interrupts ....................................................... 2370
Table 457. Response type and SDMMC_RESPxR registers .................................................. 2378
Table 458. SDMMC register map .................................................... 2394
Table 459. CAN subsystem I/O signals ............................................ 2397
Table 460. CAN triggers ............................................................. 2398
Table 461. CAN subsystem I/O pins ............................................... 2398
Table 462. Main features ........................................................... 2400
Table 463. DLC coding in FDCAN .................................................. 2405
Table 464. Example of filter configuration for Rx buffers ................................................. 2417
Table 465. Example of filter configuration for Debug messages ........................... 2418
Table 466. Possible configurations for frame transmission .................................................... 2418
Table 467. Tx buffer/FIFO - queue element size ........................................... 2419
Table 468. First byte of level 1 reference message ................................................................. 2429
Table 469. First four bytes of level 2 reference message ..................................................... 2430
Table 470. First four bytes of level 0 reference message ...................................................... 2430
Table 471. TUR configuration example ............................................................................. 2431
Table 472. System matrix, Node A .............................................................................. 2436
Table 473.Trigger list, Node A ........................................................................... 2437
Table 474. Number of data bytes transmitted with a reference message .................. 2444
Table 475. Rx buffer and FIFO element ............................................................................ 2451
Table 476. Rx buffer and FIFO element description ......................................................... 2451
Table 477. Tx buffer and FIFO element ............................................................................. 2453
Table 478. Tx buffer element description .......................................................................... 2453
Table 479. Tx Event FIFO element .................................................................................. 2455
Table 480. Tx Event FIFO element description ............................................................... 2455
Table 481. Standard message ID filter element ................................................................. 2456
Table 482. Standard message ID filter element field description ..................................... 2457
Table 483. Extended message ID filter element ................................................................. 2458
Table 484. Extended message ID filter element field description ..................................... 2458
Table 485. Trigger memory element ................................................................................. 2459
Table 486. Trigger memory element description ............................................................... 2459
Table 487. FDCAN register map and reset values ............................................................ 2518
Table 488. CCU register map and reset values ................................................................. 2528
Table 489. OTG_HS speeds supported .............................................................................. 2530
Table 490. OTG_HS implementation ............................................................................ 2532
Table 491. OTG_HS input/output pins ............................................................................. 2533
Table 492. OTG_HS input/output signals .......................................................................... 2533
Table 493. Compatibility of STM32 low power modes with the OTG .............................. 2544
Table 494. Core global control and status registers (CSRs). ............................................. 2553
Table 495. Host-mode control and status registers (CSRs) ............................................... 2554
Table 496. Device-mode control and status registers ....................................................... 2556
Table 497. Data FIFO (DFIFO) access register map ....................................................... 2558
Table 498. Power and clock gating control and status registers ....................................... 2558
Table 499. TRDT values ......................................................................................... 2567
Table 500. TRDT values (HS) .................................................................................. 2567
Table 501. Minimum duration for soft disconnect ............................................................ 2612
Table 502. OTG_HS register map and reset values .......................................................... 2640
Table 503. HDMI pin ......................................................................................... 2715
Table 504. HDMI-CEC internal input/output signals ....................................................... 2715
Table 505. Error handling timing parameters ................................................................. 2721
Table 506. TXERR timing parameters ................................................................. 2722
Table 507. HDMI-CEC interrupts ................................................................. 2723
Table 508. HDMI-CEC register map and reset values ................................... 2731
Table 509. JTAG/Serial-wire debug port pins ............................................ 2734
Table 510. Trace port pins ........................................................................... 2735
Table 511. Serial-wire trace port pins ......................................................... 2735
Table 512. Trigger pins ............................................................................... 2735
Table 513. Packet request .......................................................................... 2739
Table 514. ACK response ............................................................................ 2739
Table 515. Data transfer ............................................................................... 2739
Table 516. JTAG-DP data registers ............................................................ 2742
Table 517. Debug port registers ................................................................... 2743
Table 518. MEM-AP registers ..................................................................... 2753
Table 519. System ROM table 1 ................................................................. 2758
Table 520. System ROM table 2 ................................................................. 2758
Table 521. System ROM table register map and reset values ....................... 2764
Table 522. System CTI inputs ..................................................................... 2766
Table 523. System CTI outputs ................................................................... 2767
Table 524. Cortex-M7 CTI inputs ............................................................... 2767
Table 525. Cortex-M7 CTI outputs ............................................................. 2767
Table 526. CTI register map and reset values ............................................. 2783
Table 527. CSTF register map and reset values .......................................... 2795
Table 528. ETF register map and reset values ............................................. 2817
Table 529. TPIU register map and reset values ........................................... 2835
Table 530. SWO register map and reset values ........................................... 2848
Table 531. DBGMCU register map and reset values .................................... 2857
Table 532. Cortex-M7 CPU ROM table ...................................................... 2858
Table 533. Cortex-M7 PPB ROM table ....................................................... 2858
Table 534. Cortex-M7 CPU ROM table register map and reset values ........ 2864
Table 535. Cortex-M7 PPB ROM table register map and reset values .......... 2870
Table 536. Cortex-M7 DWT register map and reset values .......................... 2884
Table 537. Cortex-M7 ITM register map and reset values ............................ 2893
Table 538. Cortex-M7 FPB register map and reset values ............................ 2901
Table 539. Cortex-M7 ETM register map and reset values ............................ 2930
Table 540. Document revision history ........................................................ 2939
List of figures

Figure 1. System architecture for STM32H7A3/7B3/7B0xx devices ............................................. 101
Figure 2. AXI interconnect .............................................................. 106
Figure 3. RAM ECC controller implementation schematic ...................................................... 138
Figure 4. Connection between RAM ECC controller and RAMECC monitoring unit ................ 138
Figure 5. FLASH block diagram .................................................................................. 146
Figure 6. Detailed FLASH architecture ........................................................................ 148
Figure 7. Embedded Flash memory organization .............................................................. 149
Figure 8. Embedded Flash memory usage ....................................................................... 153
Figure 9. FLASH protection mechanisms ....................................................................... 155
Figure 10. FLASH read pipeline architecture ................................................................. 157
Figure 11. FLASH write pipeline architecture ................................................................. 161
Figure 12. Flash bank swapping sequence ...................................................................... 173
Figure 13. RDP protection transition scheme .................................................................. 189
Figure 14. Example of protected region overlapping ........................................................ 191
Figure 15. Flash memory areas and services in Standard and Secure access modes .......... 252
Figure 16. Bootloader state machine in Secure access mode ............................................. 253
Figure 17. Core access to Flash memory areas .................................................................. 256
Figure 18. Power control block diagram ....................................................................... 258
Figure 19. Power supply overview ................................................................................. 262
Figure 20. System supply configurations ....................................................................... 263
Figure 21. Device startup with V\textsubscript{CORE} supplied from voltage regulator .......... 266
Figure 22. Device startup with V\textsubscript{CORE} supplied from SMPS .................................... 267
Figure 23. Backup domain ............................................................................................ 272
Figure 24. USB supply configurations ........................................................................... 273
Figure 25. Power-on reset/power-down reset waveform .................................................... 274
Figure 26. BOR thresholds ............................................................................................ 275
Figure 27. PVD thresholds ............................................................................................ 276
Figure 28. AVD thresholds ............................................................................................ 276
Figure 29. V\textsubscript{BAT} thresholds .................................................................................. 277
Figure 30. Temperature thresholds ................................................................................ 279
Figure 31. V\textsubscript{CORE} overvoltage protection ............................................................... 280
Figure 32. V\textsubscript{CORE} voltage scaling versus system power modes .............................. 285
Figure 33. Power-control modes detailed state diagram .................................................... 286
Figure 34. Dynamic voltage scaling in Run mode .............................................................. 288
Figure 35. Dynamic voltage scaling behavior with CPU domain and system in Stop mode ... 290
Figure 36. Dynamic voltage scaling system Standby mode ............................................. 292
Figure 37. Dynamic voltage scaling behavior with CPU domain in DStop or DStop2 mode and SmartRun domain in Autonomous mode ...................................................... 294
Figure 38. EXTI, RCC and PWR interconnections ............................................................. 319
Figure 39. Timing diagram of SRD SRAM-to-LPUART1 transfer with BDMA2 and SRD domain in Autonomous mode ................................................................. 323
Figure 40. BDMA2 and DMAMUX2 interconnection .......................................................... 325
Figure 41. Timing diagram of LPUART1 transmission with SRD domain in Autonomous mode ........................................................................................................... 328
Figure 42. RCC block diagram ...................................................................................... 331
Figure 43. System reset circuit ....................................................................................... 334
Figure 44. Boot sequences versus system states ............................................................... 339
Figure 45. Top-level clock tree ...................................................................................... 341
List of figures

Figure 46. HSE/LSE clock source ................................................................. 342
Figure 47. HSE/LSE bypass ................................................................. 343
Figure 48. HSI calibration flow ................................................................. 346
Figure 49. CSI calibration flow ................................................................. 347
Figure 50. PLL block diagram ................................................................. 350
Figure 51. PLLs Initialization flowchart ................................................................. 353
Figure 52. Core and bus clock generation ................................................................. 355
Figure 53. Kernel clock distribution for SAI s, DFSDMs and SPDIFRX ................................................................. 361
Figure 54. Kernel clock distribution for SPI s and SPI/I2S ................................................................. 362
Figure 55. Kernel clock distribution for I2Cs ................................................................. 363
Figure 56. Kernel clock distribution for UARTs, USARTs and LPUART1 ................................................................. 364
Figure 57. Kernel clock distribution for LTDC ................................................................. 364
Figure 58. Kernel clock distribution for SDMMC, OCTOSPI and FMC ................................................................. 365
Figure 59. Kernel clock distribution for USB (2) ................................................................. 365
Figure 60. Kernel clock distribution for ADCs, SWP, RNG and FDCANs ................................................................. 366
Figure 61. Kernel clock distribution for LPTIMs and HDMI-CEC (2) ................................................................. 367
Figure 62. Peripheral allocation example ................................................................. 370
Figure 63. Kernel clock switching ................................................................. 373
Figure 64. Peripheral kernel clock enable logic details ................................................................. 375
Figure 65. Bus clock enable logic ................................................................. 380
Figure 66. CRS block diagram ................................................................. 478
Figure 67. CRS counter behavior ................................................................. 480
Figure 68. HSEM block diagram ................................................................. 490
Figure 69. Procedure state diagram ................................................................. 491
Figure 70. Interrupt state diagram ................................................................. 494
Figure 71. Basic structure of an I/O port bit ................................................................. 503
Figure 72. Basic structure of a 5-Volt tolerant I/O port bit ................................................................. 503
Figure 73. Input floating/pull up/pull down configurations ................................................................. 508
Figure 74. Output configuration ................................................................. 509
Figure 75. Alternate function configuration ................................................................. 510
Figure 76. High impedance-analog configuration ................................................................. 510
Figure 77. Analog inputs connected to ADC inputs ................................................................. 511
Figure 78. Timer break input lockup block diagram ................................................................. 522
Figure 79. I/O compensation cell block diagram ................................................................. 523
Figure 80. MDMA block diagram ................................................................. 565
Figure 81. DMA block diagram ................................................................. 594
Figure 82. Peripheral-to-memory mode ................................................................. 598
Figure 83. Memory-to-peripheral mode ................................................................. 599
Figure 84. Memory-to-memory mode ................................................................. 600
Figure 85. FIFO structure ................................................................. 605
Figure 86. BDMA block diagram ................................................................. 630
Figure 87. DMAMUX block diagram ................................................................. 658
Figure 88. Synchronization mode of the DMAMUX request line multiplexer channel ................................................................. 661
Figure 89. Event generation of the DMA request line multiplexer channel ................................................................. 661
Figure 90. DMA2D block diagram ................................................................. 676
Figure 91. Intel 8080 16-bit mode (RGB565) ................................................................. 683
Figure 92. Intel 8080 18/24-bit mode (RGB888) ................................................................. 683
Figure 93. EXTI block diagram ................................................................. 722
Figure 94. Configurable event triggering logic CPU wakeup ................................................................. 724
Figure 95. Configurable event triggering logic Any wakeup ................................................................. 726
Figure 96. Direct event triggering logic CPU wakeup ................................................................. 727
Figure 97. Direct event triggering logic Any wakeup ................................................................. 728

82/2974  RM0455 Rev 3
Figure 100. Virtual buffer ................................................................. 757
Figure 99. Graphic MMU block diagram ............................................. 756
Figure 101. Virtual buffer and physical buffer memory map .................... 758
Figure 102. MMU block diagram ......................................................... 759
Figure 103. Block validation/comparator implementation ......................... 760
Figure 104. CRC calculation unit block diagram .................................... 774
Figure 105. FMC block diagram ......................................................... 781
Figure 106. FMC memory banks (default mapping) ................................. 785
Figure 107. Mode 1 read access waveforms ......................................... 796
Figure 108. Mode 1 write access waveforms ........................................ 797
Figure 109. Mode A read access waveforms ......................................... 799
Figure 110. Mode A write access waveforms ....................................... 800
Figure 111. Mode 2 and mode B read access waveforms .......................... 802
Figure 112. Mode 2 write access waveforms ....................................... 802
Figure 113. Mode B write access waveforms ....................................... 803
Figure 114. Mode C read access waveforms ....................................... 805
Figure 115. Mode C write access waveforms ...................................... 805
Figure 116. Mode D read access waveforms ....................................... 808
Figure 117. Mode D write access waveforms ...................................... 808
Figure 118. Muxed read access waveforms ........................................ 811
Figure 119. Muxed write access waveforms ....................................... 811
Figure 120. Asynchronous wait during a read access waveforms ................. 814
Figure 121. Asynchronous wait during a write access waveforms ............... 814
Figure 122. Wait configuration waveforms ........................................ 817
Figure 123. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM) .................................................. 817
Figure 124. Synchronous multiplexed write mode waveforms - PSRAM (CRAM) .................................................. 819
Figure 125. NAND Flash controller waveforms for common memory access .... 833
Figure 126. Access to non 'CE don't care' NAND-Flash .......................... 834
Figure 127. Burst write SDRAM access waveforms ............................... 844
Figure 128. Burst read SDRAM access ............................................... 845
Figure 129. Logic diagram of Read access with RBURST bit set (CAS=2, RPIPE=0) .................................................. 846
Figure 130. Read access crossing row boundary ................................... 848
Figure 131. Write access crossing row boundary .................................. 848
Figure 132. Self-refresh mode .......................................................... 851
Figure 133. Power-down mode .......................................................... 852
Figure 134. OCTOSPI block diagram for octal communication .................. 863
Figure 135. OCTOSPI block diagram in Quad mode ................................ 863
Figure 136. OCTOSPI block diagram when dual-Flash is enabled ................. 864
Figure 137. SDR read command in Octal mode example .......................... 865
Figure 138. DTR read in Octal mode with DQS (Macronix mode) example .... 868
Figure 139. SDR write command in Octo-SPI mode example ..................... 869
Figure 140. DTR write in Octal mode (Macronix mode) example ................. 870
Figure 141. Example of HyperBus read operation .................................. 872
Figure 142. HyperBus read operation with initial latency ....................... 873
Figure 143. HyperBus write operation with initial latency ....................... 873
Figure 144. HyperBus read operation with additional latency ................... 874
Figure 145. HyperBus write operation with additional latency ................... 874
Figure 146. HyperBus write operation with no latency ........................... 875
Figure 147. HyperBus read operation page crossing with latency ............... 875
Figure 148. nCS when CKMODE=0 (T = CLK period) ............................. 887
Figure 149. CS when CKMODE=1 in SDR mode (T = CLK period) ............... 887
Figure 150. nCS when CKMODE=1 in DDTR mode (T = CLK period) ........................................ 888
Figure 151. nCS when CKMODE=1 with an abort (T = CLK period) ........................................ 888
Figure 152. OCTOSPIM block diagram ................................................................. 917
Figure 153. DLYB block diagram .............................................................. 924
Figure 154. ADC block diagram .............................................................. 932
Figure 155. ADC Clock scheme ............................................................. 935
Figure 156. ADC1 connectivity ............................................................. 936
Figure 157. ADC2 connectivity ............................................................. 937
Figure 158. ADC calibration ................................................................. 941
Figure 159. Updating the ADC offset calibration factor ............................................. 941
Figure 160. Mixing single-ended and differential channels ......................................... 942
Figure 161. Enabling / Disabling the ADC ...................................................... 945
Figure 162. Analog to digital conversion time ..................................................... 951
Figure 163. Stopping ongoing regular conversions .................................................. 952
Figure 164. Stopping ongoing regular and injected conversions ................................. 952
Figure 165. Triggers are shared between ADC master and ADC slave .......................... 954
Figure 166. Injected conversion latency ............................................................. 957
Figure 167. Example of JSQR queue of context (sequence change) ............................... 960
Figure 168. Example of JSQR queue of context (trigger change) ................................. 961
Figure 169. Example of JSQR queue of context with overflow before conversion ............ 961
Figure 170. Example of JSQR queue of context with overflow during conversion .............. 962
Figure 171. Example of JSQR queue of context with empty queue (case JQM=0) .......... 962
Figure 172. Example of JSQR queue of context with empty queue (case JQM=1) .......... 963
Figure 173. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion ............................................ 963
Figure 174. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion and a new trigger occurs ............................................................. 964
Figure 175. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs outside an ongoing conversion ........................................... 964
Figure 176. Flushing JSQR queue of context by setting JADSTP=1 (JQM=1) ................. 965
Figure 177. Flushing JSQR queue of context by setting ADDIS=1 (JQM=0) .................. 965
Figure 178. Flushing JSQR queue of context by setting ADDIS=1 (JQM=1) .................. 966
Figure 179. Single conversions of a sequence, software trigger .................................. 968
Figure 180. Continuous conversion of a sequence, software trigger .............................. 968
Figure 181. Single conversions of a sequence, hardware trigger .................................. 969
Figure 182. Continuous conversions of a sequence, hardware trigger ............................ 969
Figure 183. Right alignment (offset disabled, unsigned value) ..................................... 971
Figure 184. Right alignment (offset enabled, signed value) ....................................... 971
Figure 185. Left alignment (offset disabled, unsigned value) ..................................... 972
Figure 186. Left alignment (offset enabled, signed value) ....................................... 972
Figure 187. Example of overrun (OVR = 1 and OVRMOD = 1) .................................... 974
Figure 188. AUTDLY=1, regular conversion in continuous mode, software trigger .......... 978
Figure 189. AUTDLY=1, regular HW conversions interrupted by injected conversions (DISCEN=0; JDISCEN=0) ................................................................. 978
Figure 190. AUTDLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1; JDISCEN=1) ................................................................. 979
Figure 191. AUTDLY=1, regular continuous conversions interrupted by injected conversions ................................................................. 980
Figure 192. AUTDLY=1 in auto- injected mode (JAUTO=1) ........................................ 980
Figure 193. Analog watchdog guarded area .......................................................... 981
Figure 194. ADCy_AWDx_OUT signal generation (on all regular channels) .................. 983
Figure 195. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by SW) .......... 983
Figure 196. ADCy_AWDx_OUT signal generation (on a single regular channel) .......................... 984
Figure 197. ADCy_AWDx_OUT signal generation (on all injected channels) .......................... 984
Figure 198. 16-bit result oversampling with 10-bits right shift and rounding .......................... 985
Figure 199. Triggered regular oversampling mode (TROVS bit = 1) ...................................... 987
Figure 200. Regular oversampling modes (4x ratio) ............................................................. 988
Figure 201. Regular and injected oversampling modes used simultaneously ......................... 988
Figure 202. Triggered regular oversampling with injection .................................................. 989
Figure 203. Oversampling in auto-injected mode ................................................................. 988
Figure 204. Dual ADC block diagram(1) .............................................................................. 992
Figure 205. Injected simultaneous mode on 4 channels: dual ADC mode ............................... 993
Figure 206. Regular simultaneous mode on 16 channels: dual ADC mode ............................. 995
Figure 207. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode ...... 996
Figure 208. Interleaved mode on 1 channel in single conversion mode: dual ADC mode .......... 997
Figure 209. Interleaved conversion with injection ................................................................. 997
Figure 210. Alternate trigger: injected group of each ADC .................................................... 998
Figure 211. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode .......... 999
Figure 212. Alternate + regular simultaneous .................................................................... 1000
Figure 213. Case of trigger occurring during injected conversion ......................................... 1000
Figure 214. Interleaved single channel CH0 with injected sequence CH11, CH12 .................. 1001
Figure 215. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12  
- case 1: Master interrupted first .................................................................................. 1001
Figure 216. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12  
- case 2: Slave interrupted first .................................................................................. 1001
Figure 217. DMA Requests in regular simultaneous mode when DAMDF=0b00 ................... 1002
Figure 218. DMA requests in regular simultaneous mode when DAMDF=0b10 .................... 1003
Figure 219. DMA requests in interleaved mode when DAMDF=0b10 ................................. 1003
Figure 220. Temperature sensor channel block diagram .................................................... 1005
Figure 221. VBAT channel block diagram ...................................................................... 1007
Figure 222. VREFINT channel block diagram .................................................................. 1007
Figure 223. Temperature sensor functional block diagram .............................................. 1055
Figure 224. Method for low REF_CLK frequencies ......................................................... 1057
Figure 225. Method for high REF_CLK frequencies ......................................................... 1057
Figure 226. Temperature sensor sequence .................................................................... 1060
Figure 227. Dual-channel DAC block diagram ................................................................. 1073
Figure 228. Data registers in single DAC channel mode ..................................................... 1076
Figure 229. Data registers in dual DAC channel mode ........................................................ 1077
Figure 230. Timing diagram for conversion with trigger disabled TEN = 0 ......................... 1077
Figure 231. DAC LFSR register calculation algorithm ..................................................... 1079
Figure 232. DAC conversion (SW trigger enabled) with LFSR wave generation ................. 1079
Figure 233. DAC triangle wave generation ..................................................................... 1080
Figure 234. DAC conversion (SW trigger enabled) with triangle wave generation ............ 1080
Figure 235. DAC Sample and hold mode phase diagram .................................................. 1083
Figure 236. Comparator block diagram ....................................................................... 1113
Figure 237. Comparator hysteresis ................................................................................... 1116
Figure 238. Comparator output blanking ........................................................................... 1117
Figure 239. Output redirection ....................................................................................... 1119
Figure 240. Scaler block diagram ................................................................................... 1120
Figure 241. Standalone mode: external gain setting mode .............................................. 1133
Figure 242. Follower configuration .................................................................................. 1134
Figure 243. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used ........ 1135
Figure 244. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering. 1136
<table>
<thead>
<tr>
<th>Figure No.</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>347</td>
<td>Output compare mode, toggle on OC1</td>
<td>1467</td>
</tr>
<tr>
<td>348</td>
<td>Edge-aligned PWM waveforms (ARR=8)</td>
<td>1468</td>
</tr>
<tr>
<td>349</td>
<td>Center-aligned PWM waveforms (ARR=8)</td>
<td>1469</td>
</tr>
<tr>
<td>350</td>
<td>Generation of 2 phase-shifted PWM signals with 50% duty cycle</td>
<td>1471</td>
</tr>
<tr>
<td>351</td>
<td>Combined PWM mode on channel 1 and 3</td>
<td>1472</td>
</tr>
<tr>
<td>352</td>
<td>3-phase combined PWM signals with multiple trigger pulses per period</td>
<td>1473</td>
</tr>
<tr>
<td>353</td>
<td>Complementary output with dead-time insertion</td>
<td>1474</td>
</tr>
<tr>
<td>354</td>
<td>Dead-time waveforms with delay greater than the negative pulse</td>
<td>1474</td>
</tr>
<tr>
<td>355</td>
<td>Dead-time waveforms with delay greater than the positive pulse</td>
<td>1475</td>
</tr>
<tr>
<td>356</td>
<td>Break and Break2 circuitry overview</td>
<td>1477</td>
</tr>
<tr>
<td>357</td>
<td>Various output behavior in response to a break event on BRK (OSSI = 1)</td>
<td>1479</td>
</tr>
<tr>
<td>358</td>
<td>PWM output state following BRK and BRK2 pins assertion (OSSI=1)</td>
<td>1480</td>
</tr>
<tr>
<td>359</td>
<td>PWM output state following BRK assertion (OSSI=0)</td>
<td>1481</td>
</tr>
<tr>
<td>360</td>
<td>Output redirection (BRK2 request not represented)</td>
<td>1482</td>
</tr>
<tr>
<td>361</td>
<td>Clearing TIMx OCxREF</td>
<td>1483</td>
</tr>
<tr>
<td>362</td>
<td>6-step generation, COM example (OSSR=1)</td>
<td>1484</td>
</tr>
<tr>
<td>363</td>
<td>Example of one pulse mode</td>
<td>1485</td>
</tr>
<tr>
<td>364</td>
<td>Retriggerable one pulse mode</td>
<td>1487</td>
</tr>
<tr>
<td>365</td>
<td>Example of counter operation in encoder interface mode.</td>
<td>1488</td>
</tr>
<tr>
<td>366</td>
<td>Example of encoder interface mode with T11FP1 polarity inverted.</td>
<td>1489</td>
</tr>
<tr>
<td>367</td>
<td>Measuring time interval between edges on 3 signals</td>
<td>1490</td>
</tr>
<tr>
<td>368</td>
<td>Example of Hall sensor interface</td>
<td>1492</td>
</tr>
<tr>
<td>369</td>
<td>Control circuit in reset mode</td>
<td>1493</td>
</tr>
<tr>
<td>370</td>
<td>Control circuit in Gated mode</td>
<td>1494</td>
</tr>
<tr>
<td>371</td>
<td>Control circuit in trigger mode</td>
<td>1495</td>
</tr>
<tr>
<td>372</td>
<td>Control circuit in external clock mode 2 + trigger mode</td>
<td>1496</td>
</tr>
<tr>
<td>373</td>
<td>General-purpose timer block diagram</td>
<td>1546</td>
</tr>
<tr>
<td>374</td>
<td>Counter timing diagram with prescaler division change from 1 to 2</td>
<td>1548</td>
</tr>
<tr>
<td>375</td>
<td>Counter timing diagram with prescaler division change from 1 to 4</td>
<td>1548</td>
</tr>
<tr>
<td>376</td>
<td>Counter timing diagram, internal clock divided by 1</td>
<td>1549</td>
</tr>
<tr>
<td>377</td>
<td>Counter timing diagram, internal clock divided by 2</td>
<td>1550</td>
</tr>
<tr>
<td>378</td>
<td>Counter timing diagram, internal clock divided by 4</td>
<td>1550</td>
</tr>
<tr>
<td>379</td>
<td>Counter timing diagram, internal clock divided by N</td>
<td>1551</td>
</tr>
<tr>
<td>380</td>
<td>Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)</td>
<td>1551</td>
</tr>
<tr>
<td>381</td>
<td>Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)</td>
<td>1552</td>
</tr>
<tr>
<td>382</td>
<td>Counter timing diagram, internal clock divided by 1</td>
<td>1553</td>
</tr>
<tr>
<td>383</td>
<td>Counter timing diagram, internal clock divided by 2</td>
<td>1553</td>
</tr>
<tr>
<td>384</td>
<td>Counter timing diagram, internal clock divided by 4</td>
<td>1554</td>
</tr>
<tr>
<td>385</td>
<td>Counter timing diagram, internal clock divided by N</td>
<td>1554</td>
</tr>
<tr>
<td>386</td>
<td>Counter timing diagram, Update event when repetition counter is not used</td>
<td>1555</td>
</tr>
<tr>
<td>387</td>
<td>Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6</td>
<td>1556</td>
</tr>
<tr>
<td>388</td>
<td>Counter timing diagram, internal clock divided by 2</td>
<td>1557</td>
</tr>
<tr>
<td>389</td>
<td>Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36</td>
<td>1557</td>
</tr>
<tr>
<td>390</td>
<td>Counter timing diagram, internal clock divided by N</td>
<td>1558</td>
</tr>
<tr>
<td>391</td>
<td>Counter timing diagram, Update event with ARPE=1 (counter underflow)</td>
<td>1558</td>
</tr>
<tr>
<td>392</td>
<td>Counter timing diagram, Update event with ARPE=1 (counter overflow)</td>
<td>1559</td>
</tr>
<tr>
<td>393</td>
<td>Control circuit in normal mode, internal clock divided by 1</td>
<td>1560</td>
</tr>
<tr>
<td>394</td>
<td>TI2 external clock connection example</td>
<td>1560</td>
</tr>
<tr>
<td>395</td>
<td>Control circuit in external clock mode 1</td>
<td>1561</td>
</tr>
<tr>
<td>396</td>
<td>External trigger input block</td>
<td>1562</td>
</tr>
<tr>
<td>397</td>
<td>Control circuit in external clock mode 2</td>
<td>1563</td>
</tr>
</tbody>
</table>
Figure 398. Capture/Compare channel (example: channel 1 input stage) .............................. 1563
Figure 399. Capture/Compare channel 1 main circuit ......................................................... 1564
Figure 400. Output stage of Capture/Compare channel (channel 1) ...................................... 1564
Figure 401. PWM input mode timing ..................................................................................... 1566
Figure 402. Output compare mode, toggle on OC1. ............................................................... 1568
Figure 403. Edge-aligned PWM waveforms (ARR=8) ............................................................. 1569
Figure 404. Center-aligned PWM waveforms (ARR=8) ......................................................... 1571
Figure 405. Generation of 2 phase-shifted PWM signals with 50% duty cycle ......................... 1572
Figure 406. Combined PWM mode on channels 1 and 3 ........................................................ 1573
Figure 407. Clearing TIMx OCxREF ...................................................................................... 1574
Figure 408. Example of one-pulse mode. .............................................................................. 1575
Figure 409. Retriggerable one-pulse mode. ......................................................................... 1577
Figure 410. Example of counter operation in encoder interface mode ................................. 1578
Figure 411. Example of encoder interface mode with TI1FP1 polarity inverted ..................... 1579
Figure 412. Control circuit in reset mode ............................................................................... 1580
Figure 413. Control circuit in gated mode ............................................................................. 1581
Figure 414. Control circuit in trigger mode. ......................................................................... 1582
Figure 415. Control circuit in external clock mode 2 + trigger mode .................................... 1583
Figure 416. Master/Slave timer example ............................................................................. 1583
Figure 417. Gating TIM2 with OC1REF of TIM3 ................................................................. 1584
Figure 418. Gating TIM2 with Enable of TIM3 ..................................................................... 1585
Figure 419. Triggering TIM2 with update of TIM3 ............................................................... 1586
Figure 420. Triggering TIM2 with Enable of TIM3 ............................................................... 1586
Figure 421. Triggering TIM3 and TIM2 with TIM3 TI1 input. ............................................... 1587
Figure 422. General-purpose timer block diagram (TIM12). .................................................. 1621
Figure 423. General-purpose timer block diagram (TIM13/TIM14) ...................................... 1622
Figure 424. Counter timing diagram with prescaler division change from 1 to 2 ..................... 1624
Figure 425. Counter timing diagram with prescaler division change from 1 to 4 ..................... 1624
Figure 426. Counter timing diagram, internal clock divided by 1 ............................................ 1625
Figure 427. Counter timing diagram, internal clock divided by 2 ............................................ 1626
Figure 428. Counter timing diagram, internal clock divided by 4 ............................................ 1626
Figure 429. Counter timing diagram, internal clock divided by N .......................................... 1627
Figure 430. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) ........................................................................................................... 1627
Figure 431. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) ........................................................................................................... 1628
Figure 432. Control circuit in normal mode, internal clock divided by 1 ................................ 1629
Figure 433. TI2 external clock connection example. .............................................................. 1629
Figure 434. Control circuit in external clock mode 1 ............................................................. 1630
Figure 435. Capture/compare channel (example: channel 1 input stage) .............................. 1631
Figure 436. Capture/compare channel 1 main circuit ............................................................. 1631
Figure 437. Output stage of capture/compare channel (channel 1). ....................................... 1632
Figure 438. PWM input mode timing ..................................................................................... 1634
Figure 439. Output compare mode, toggle on OC1. ............................................................... 1636
Figure 440. Edge-aligned PWM waveforms (ARR=8) ............................................................. 1637
Figure 441. Combined PWM mode on channel 1 and 2 ........................................................ 1638
Figure 442. Example of one pulse mode. .............................................................................. 1639
Figure 443. Retriggerable one pulse mode .......................................................................... 1640
Figure 444. Measuring time interval between edges on 2 signals ........................................... 1641
Figure 445. Control circuit in reset mode ............................................................................. 1642
Figure 446. Control circuit in gated mode ............................................................................. 1643
Figure 447. Control circuit in trigger mode .......................................................................... 1644
Figure 448. TIM15 block diagram ................................................................. 1674
Figure 449. TIM16/TIM17 block diagram ..................................................... 1675
Figure 450. Counter timing diagram with prescaler division change from 1 to 2 ........ 1677
Figure 451. Counter timing diagram with prescaler division change from 1 to 4 ........ 1677
Figure 452. Counter timing diagram, internal clock divided by 1 ....................... 1679
Figure 453. Counter timing diagram, internal clock divided by 2 ....................... 1679
Figure 454. Counter timing diagram, internal clock divided by 4 ....................... 1680
Figure 455. Counter timing diagram, internal clock divided by N ...................... 1680
Figure 456. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) ................................................................. 1681
Figure 457. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) ................................................................. 1681
Figure 458. Update rate examples depending on mode and TIMx_RCR register settings 1683
Figure 459. Control circuit in normal mode, internal clock divided by 1 ............... 1684
Figure 460. T12 external clock connection example ......................................... 1684
Figure 461. Control circuit in external clock mode 1 ....................................... 1685
Figure 462. Capture/compare channel (example: channel 1 input stage) .............. 1686
Figure 463. Capture/compare channel 1 main circuit ...................................... 1686
Figure 464. Output stage of capture/compare channel (channel 1) ...................... 1687
Figure 465. Output stage of capture/compare channel (channel 2 for TIM15) ........ 1687
Figure 466. PWM input mode timing .......................................................... 1689
Figure 467. Output compare mode, toggle on OC1. ....................................... 1691
Figure 468. Edge-aligned PWM waveforms (ARR=8) .................................... 1692
Figure 469. Combined PWM mode on channel 1 and 2 .................................. 1693
Figure 470. Complementary output with dead-time insertion ............................ 1694
Figure 471. Dead-time waveforms with delay greater than the negative pulse ....... 1694
Figure 472. Dead-time waveforms with delay greater than the positive pulse ....... 1695
Figure 473. Break circuitry overview .......................................................... 1697
Figure 474. Output behavior in response to a break ....................................... 1699
Figure 475. Output redirection .................................................................... 1701
Figure 476. Example of one pulse mode ....................................................... 1703
Figure 477. Retriggerable one pulse mode ..................................................... 1704
Figure 478. Measuring time interval between edges on 2 signals ....................... 1706
Figure 479. Control circuit in reset mode ...................................................... 1707
Figure 480. Control circuit in gated mode ..................................................... 1708
Figure 481. Control circuit in trigger mode .................................................... 1709
Figure 482. Basic timer block diagram ......................................................... 1761
Figure 483. Counter timing diagram with prescaler division change from 1 to 2 .... 1763
Figure 484. Counter timing diagram with prescaler division change from 1 to 4 .... 1763
Figure 485. Counter timing diagram, internal clock divided by 1 ....................... 1764
Figure 486. Counter timing diagram, internal clock divided by 2 ....................... 1765
Figure 487. Counter timing diagram, internal clock divided by 4 ....................... 1765
Figure 488. Counter timing diagram, internal clock divided by N ...................... 1766
Figure 489. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) ................................................................. 1766
Figure 490. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) ................................................................. 1767
Figure 491. Control circuit in normal mode, internal clock divided by 1 ............... 1768
Figure 492. Low-power timer block diagram (LPTIM1 and LPTIM2) ............... 1775
Figure 493. Low-power timer block diagram (LPTIM3) .................................. 1776
Figure 494. Glitch filter timing diagram ....................................................... 1780
Figure 495. LPTIM output waveform, single counting mode configuration ........ 1781
Figure 545. Mute mode using address mark detection .................................................. 1976
Figure 546. Break detection in LIN mode (11-bit break length - LBDL bit is set) .......... 1979
Figure 547. Break detection in LIN mode vs. Framing error detection. ...................... 1980
Figure 548. USART example of synchronous master transmission. ............................. 1981
Figure 549. USART data clock timing diagram in synchronous master mode
(M bits =00) .............................................................................................................. 1981
Figure 550. USART data clock timing diagram in synchronous master mode
(M bits = 01) ............................................................................................................ 1982
Figure 551. USART data clock timing diagram in synchronous slave mode
(M bits =00) .............................................................................................................. 1983
Figure 552. ISO 7816-3 asynchronous protocol .......................................................... 1985
Figure 553. Parity error detection using the 1.5 stop bits ............................................ 1987
Figure 554. IrDA SIR ENDEC block diagram. ............................................................ 1991
Figure 555. IrDA data modulation (3/16) - Normal mode ........................................ 1991
Figure 556. Transmission using DMA ....................................................................... 1993
Figure 557. Reception using DMA .............................................................................. 1994
Figure 558. Hardware flow control between 2 USARTs ............................................ 1994
Figure 559. RS232 RTS flow control ............................................................................ 1995
Figure 560. RS232 CTS flow control .......................................................................... 1996
Figure 561. Wakeup event verified (wakeup event = address match, FIFO disabled) ..... 1999
Figure 562. Wakeup event not verified (wakeup event = address match,
FIFO disabled) ........................................................................................................ 1999
Figure 563. LPUART block diagram .......................................................................... 2042
Figure 564. LPUART word length programming ....................................................... 2044
Figure 565. Configurable stop bits .............................................................................. 2046
Figure 566. TC/TXE behavior when transmitting ...................................................... 2048
Figure 567. Ipuart_ker_ck clock divider block diagram ............................................. 2051
Figure 568. Mute mode using Idle line detection ....................................................... 2055
Figure 569. Mute mode using address mark detection .............................................. 2056
Figure 570. Transmission using DMA ........................................................................ 2058
Figure 571. Reception using DMA .............................................................................. 2059
Figure 572. Hardware flow control between 2 LPUARTs ........................................... 2060
Figure 573. RS232 RTS flow control .......................................................................... 2060
Figure 574. RS232 CTS flow control .......................................................................... 2061
Figure 575. Wakeup event verified (wakeup event = address match,
FIFO disabled) ........................................................................................................ 2064
Figure 576. Wakeup event not verified (wakeup event = address match,
FIFO disabled) ........................................................................................................ 2064
Figure 577. SPI2S block diagram .............................................................................. 2094
Figure 578. Full-duplex single master/ single slave application ................................. 2096
Figure 579. Half-duplex single master/ single slave application ............................... 2096
Figure 580. Simplex single master/single slave application (master in transmit-only/
slave in receive-only mode) .................................................................................... 2098
Figure 581. Master and three independent slaves at star topology ............................. 2099
Figure 582. Master and three slaves at circular (daisy chain) topology ..................... 2100
Figure 583. Multi-master application .......................................................................... 2101
Figure 584. Scheme of SS control logic ..................................................................... 2103
Figure 585. Data flow timing control (SSOE=1, SSOM=0, SSM=0) ......................... 2103
Figure 586. SS interleaving pulses between data (SSOE=1, SSOM=1,SSM=0). ......... 2104
Figure 587. Data clock timing diagram ...................................................................... 2106
Figure 588. Data alignment when data size is not equal to 8-bit, 16-bit or 32-bit ....... 2107
Figure 589. Packing data in FIFO for transmission and reception ........................... 2114
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>590.</td>
<td>TI mode transfer</td>
<td>2116</td>
</tr>
<tr>
<td>591.</td>
<td>Low-power mode application example</td>
<td>2121</td>
</tr>
<tr>
<td>592.</td>
<td>Waveform examples</td>
<td>2127</td>
</tr>
<tr>
<td>593.</td>
<td>Master I2S Philips protocol waveforms (16/32-bit full accuracy)</td>
<td>2128</td>
</tr>
<tr>
<td>594.</td>
<td>I2S Philips standard waveforms</td>
<td>2128</td>
</tr>
<tr>
<td>595.</td>
<td>Master MSB Justified 16-bit or 32-bit full-accuracy length</td>
<td>2129</td>
</tr>
<tr>
<td>596.</td>
<td>Master MSB justified 16 or 24-bit data length</td>
<td>2129</td>
</tr>
<tr>
<td>597.</td>
<td>Slave MSB justified</td>
<td>2130</td>
</tr>
<tr>
<td>598.</td>
<td>LSB justified 16 or 24-bit data length</td>
<td>2130</td>
</tr>
<tr>
<td>599.</td>
<td>Master PCM when the frame length is equal the data length</td>
<td>2131</td>
</tr>
<tr>
<td>600.</td>
<td>Master PCM standard waveforms</td>
<td>2131</td>
</tr>
<tr>
<td>601.</td>
<td>Slave PCM waveforms</td>
<td>2132</td>
</tr>
<tr>
<td>602.</td>
<td>Start-up sequence, I2S Philips standard, master</td>
<td>2134</td>
</tr>
<tr>
<td>603.</td>
<td>Start-up sequence, I2S Philips standard, slave</td>
<td>2135</td>
</tr>
<tr>
<td>604.</td>
<td>Stop sequence, I2S Philips standard, master</td>
<td>2136</td>
</tr>
<tr>
<td>605.</td>
<td>I2S clock generator architecture</td>
<td>2136</td>
</tr>
<tr>
<td>606.</td>
<td>Data Format</td>
<td>2139</td>
</tr>
<tr>
<td>607.</td>
<td>Handling of underrun situation</td>
<td>2140</td>
</tr>
<tr>
<td>608.</td>
<td>Handling of overrun situation</td>
<td>2141</td>
</tr>
<tr>
<td>609.</td>
<td>Frame error detection, with FIXCH=0</td>
<td>2142</td>
</tr>
<tr>
<td>610.</td>
<td>Frame error detection, with FIXCH=1</td>
<td>2142</td>
</tr>
<tr>
<td>611.</td>
<td>SAI functional block diagram</td>
<td>2169</td>
</tr>
<tr>
<td>612.</td>
<td>Audio frame</td>
<td>2173</td>
</tr>
<tr>
<td>613.</td>
<td>FS role is start of frame + channel side identification (FSDEF = TRIS = 1)</td>
<td>2175</td>
</tr>
<tr>
<td>614.</td>
<td>FS role is start of frame (FSDEF = 0)</td>
<td>2176</td>
</tr>
<tr>
<td>615.</td>
<td>Slot size configuration with FBOFF = 0 in SAI_xSLOTTR</td>
<td>2177</td>
</tr>
<tr>
<td>616.</td>
<td>First bit offset</td>
<td>2177</td>
</tr>
<tr>
<td>617.</td>
<td>Audio block clock generator overview</td>
<td>2179</td>
</tr>
<tr>
<td>618.</td>
<td>PDM typical connection and timing</td>
<td>2183</td>
</tr>
<tr>
<td>619.</td>
<td>Detailed PDM interface block diagram</td>
<td>2184</td>
</tr>
<tr>
<td>620.</td>
<td>Start-up sequence</td>
<td>2185</td>
</tr>
<tr>
<td>621.</td>
<td>SAI_ADR format in TDM, 32-bit slot width</td>
<td>2186</td>
</tr>
<tr>
<td>622.</td>
<td>SAI_ADR format in TDM, 16-bit slot width</td>
<td>2187</td>
</tr>
<tr>
<td>623.</td>
<td>SAI_ADR format in TDM, 8-bit slot width</td>
<td>2188</td>
</tr>
<tr>
<td>624.</td>
<td>AC’97 audio frame</td>
<td>2191</td>
</tr>
<tr>
<td>625.</td>
<td>Example of typical AC’97 configuration on devices featuring at least 2 embedded SAIIs (three external AC’97 decoders)</td>
<td>2192</td>
</tr>
<tr>
<td>626.</td>
<td>SPDIF format</td>
<td>2193</td>
</tr>
<tr>
<td>627.</td>
<td>SAI_xDR register ordering</td>
<td>2194</td>
</tr>
<tr>
<td>628.</td>
<td>Data companding hardware in an audio block in the SAI</td>
<td>2198</td>
</tr>
<tr>
<td>629.</td>
<td>Tristate strategy on SD output line on an inactive slot</td>
<td>2199</td>
</tr>
<tr>
<td>630.</td>
<td>FIFO underrun event</td>
<td>2200</td>
</tr>
<tr>
<td>631.</td>
<td>Overrun detection error</td>
<td>2201</td>
</tr>
<tr>
<td>632.</td>
<td>S/PDIF Sub-frame format</td>
<td>2201</td>
</tr>
<tr>
<td>633.</td>
<td>S/PDIF block format</td>
<td>2201</td>
</tr>
<tr>
<td>634.</td>
<td>S/PDIF Sub-frame format</td>
<td>2207</td>
</tr>
<tr>
<td>635.</td>
<td>S/PDIF Sub-frame format</td>
<td>2207</td>
</tr>
<tr>
<td>636.</td>
<td>S/PDIF Sub-frame format</td>
<td>2209</td>
</tr>
<tr>
<td>637.</td>
<td>Channel coding example</td>
<td>2209</td>
</tr>
<tr>
<td>638.</td>
<td>SPDIFRX decoder</td>
<td>2241</td>
</tr>
<tr>
<td>639.</td>
<td>Noise filtering and edge detection</td>
<td>2241</td>
</tr>
<tr>
<td>640.</td>
<td>Thresholds</td>
<td>2243</td>
</tr>
</tbody>
</table>
List of figures

Figure 641. Synchronization flowchart ........................................ 2245
Figure 642. Synchronization process scheduling ......................... 2246
Figure 643. SPDIFRX States ................................................. 2247
Figure 644. SPDIFRX_FMTx_DR register format ....................... 2249
Figure 645. Channel/user data format .................................. 2250
Figure 646. S/PDIF overrun error when RXSTEO = 0 ................ 2252
Figure 647. S/PDIF overrun error when RXSTEO = 1 .............. 2253
Figure 648. SPDIFRX interface interrupt mapping diagram ........ 2256
Figure 649. S1 signal coding .............................................. 2273
Figure 650. S2 signal coding .............................................. 2273
Figure 651. SWPMI block diagram ....................................... 2275
Figure 652. SWP bus states .............................................. 2278
Figure 653. SWP frame structure ....................................... 2279
Figure 654. SWPMI No software buffer mode transmission ....... 2280
Figure 655. SWPMI No software buffer mode transmission, consecutive frames 2281
Figure 656. SWPMI Multi software buffer mode transmission .... 2283
Figure 657. SWPMI No software buffer mode reception .......... 2285
Figure 658. SWPMI single software buffer mode reception ....... 2286
Figure 659. SWPMI Multi software buffer mode reception ....... 2288
Figure 660. SWPMI single buffer mode reception with CRC error .. 2289
Figure 661. MDIOS block diagram .................................... 2302
Figure 662. MDIO protocol write frame waveform .................. 2303
Figure 663. MDIO protocol read frame waveform .................... 2303
Figure 664. SDMMC “no response” and “no data” operations .... 2316
Figure 665. SDMMC (multiple) block read operation ............. 2316
Figure 666. SDMMC (multiple) block write operation ............. 2317
Figure 667. SDMMC (sequential) stream read operation ........ 2317
Figure 668. SDMMC (sequential) stream write operation ......... 2317
Figure 669. SDMMC block diagram .................................... 2319
Figure 670. SDMMC Command and data phase relation .......... 2321
Figure 671. Control unit ............................................... 2323
Figure 672. Command/Response path ................................ 2324
Figure 673. Command path state machine (CPM) ................. 2325
Figure 674. Data path .................................................. 2331
Figure 675. DDR mode data packet clocking .......................... 2332
Figure 676. DDR mode CRC status / boot acknowledgment clocking .... 2332
Figure 677. Data path state machine (DPSM) ....................... 2333
Figure 678. CLKMUX unit ............................................. 2343
Figure 679. Hardware flow timing .................................... 2347
Figure 680. Asynchronous interrupt generation .................... 2349
Figure 681. Synchronous interrupt period data read .............. 2349
Figure 682. Synchronous interrupt period data write ............ 2350
Figure 683. Asynchronous interrupt period data read .......... 2351
Figure 684. Asynchronous interrupt period data write .......... 2351
Figure 685. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25 2354
Figure 686. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104 ... 2354
Figure 687. ReadWait with SDMMC_CK < 50 MHz .............. 2355
Figure 688. ReadWait with SDMMC_CK > 50 MHz .............. 2356
Figure 689. CMD12 stream timing .................................. 2358
Figure 690. CMD5 Sleep Awake procedure ......................... 2360
Figure 691. Normal boot mode operation ........................... 2362
Figure 692. Alternative boot mode operation ..................... 2363
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>693</td>
<td>Command response R1b busy signaling</td>
<td>2364</td>
</tr>
<tr>
<td>694</td>
<td>SDMMC state control</td>
<td>2365</td>
</tr>
<tr>
<td>695</td>
<td>Card cycle power / power up diagram</td>
<td>2366</td>
</tr>
<tr>
<td>696</td>
<td>CMD11 signal voltage switch sequence</td>
<td>2367</td>
</tr>
<tr>
<td>697</td>
<td>Voltage switch transceiver typical application</td>
<td>2369</td>
</tr>
<tr>
<td>698</td>
<td>CAN subsystem</td>
<td>2399</td>
</tr>
<tr>
<td>699</td>
<td>FDCAN block diagram</td>
<td>2401</td>
</tr>
<tr>
<td>700</td>
<td>Transceiver delay measurement</td>
<td>2406</td>
</tr>
<tr>
<td>701</td>
<td>Pin control in bus monitoring mode</td>
<td>2408</td>
</tr>
<tr>
<td>702</td>
<td>Pin control in loop back mode</td>
<td>2410</td>
</tr>
<tr>
<td>703</td>
<td>Message RAM configuration</td>
<td>2411</td>
</tr>
<tr>
<td>704</td>
<td>Standard message ID filter path</td>
<td>2414</td>
</tr>
<tr>
<td>705</td>
<td>Extended message ID filter path</td>
<td>2415</td>
</tr>
<tr>
<td>706</td>
<td>Example of mixed configuration dedicated Tx buffers / Tx FIFO</td>
<td>2421</td>
</tr>
<tr>
<td>707</td>
<td>Example of mixed configuration dedicated Tx buffers / Tx queue</td>
<td>2421</td>
</tr>
<tr>
<td>708</td>
<td>Bit timing</td>
<td>2423</td>
</tr>
<tr>
<td>709</td>
<td>Bypass operation</td>
<td>2425</td>
</tr>
<tr>
<td>710</td>
<td>FSM calibration</td>
<td>2426</td>
</tr>
<tr>
<td>711</td>
<td>Cycle time and global time synchronization</td>
<td>2441</td>
</tr>
<tr>
<td>712</td>
<td>TTCAN level 0 and level 2 drift compensation</td>
<td>2442</td>
</tr>
<tr>
<td>713</td>
<td>Level 0 schedule synchronization state machine</td>
<td>2449</td>
</tr>
<tr>
<td>714</td>
<td>Level 0 master to slave relation</td>
<td>2450</td>
</tr>
<tr>
<td>715</td>
<td>OTG_HS high-speed block diagram</td>
<td>2532</td>
</tr>
<tr>
<td>716</td>
<td>SOF connectivity (SOF trigger output to TIM and ITR1 connection)</td>
<td>2543</td>
</tr>
<tr>
<td>717</td>
<td>Updating OTG_HFIR dynamically (RLDCTRL = 1)</td>
<td>2546</td>
</tr>
<tr>
<td>718</td>
<td>Device-mode FIFO address mapping and AHB FIFO access mapping</td>
<td>2547</td>
</tr>
<tr>
<td>719</td>
<td>Host-mode FIFO address mapping and AHB FIFO access mapping</td>
<td>2548</td>
</tr>
<tr>
<td>720</td>
<td>Interrupt hierarchy</td>
<td>2552</td>
</tr>
<tr>
<td>721</td>
<td>Transmit FIFO write task</td>
<td>2656</td>
</tr>
<tr>
<td>722</td>
<td>Receive FIFO read task</td>
<td>2657</td>
</tr>
<tr>
<td>723</td>
<td>Normal bulk/control OUT/SETUP</td>
<td>2658</td>
</tr>
<tr>
<td>724</td>
<td>Bulk/control IN transactions</td>
<td>2662</td>
</tr>
<tr>
<td>725</td>
<td>Normal interrupt OUT</td>
<td>2665</td>
</tr>
<tr>
<td>726</td>
<td>Normal interrupt IN</td>
<td>2670</td>
</tr>
<tr>
<td>727</td>
<td>Isochronous OUT transactions</td>
<td>2672</td>
</tr>
<tr>
<td>728</td>
<td>Isochronous IN transactions</td>
<td>2675</td>
</tr>
<tr>
<td>729</td>
<td>Normal bulk/control OUT/SETUP transactions - DMA</td>
<td>2677</td>
</tr>
<tr>
<td>730</td>
<td>Normal bulk/control IN transaction - DMA</td>
<td>2679</td>
</tr>
<tr>
<td>731</td>
<td>Normal interrupt OUT transactions - DMA mode</td>
<td>2680</td>
</tr>
<tr>
<td>732</td>
<td>Normal interrupt IN transactions - DMA mode</td>
<td>2681</td>
</tr>
<tr>
<td>733</td>
<td>Normal isochronous OUT transaction - DMA mode</td>
<td>2682</td>
</tr>
<tr>
<td>734</td>
<td>Normal isochronous IN transactions - DMA mode</td>
<td>2683</td>
</tr>
<tr>
<td>735</td>
<td>Receive FIFO packet read</td>
<td>2689</td>
</tr>
<tr>
<td>736</td>
<td>Processing a SETUP packet</td>
<td>2691</td>
</tr>
<tr>
<td>737</td>
<td>Bulk OUT transaction</td>
<td>2698</td>
</tr>
<tr>
<td>738</td>
<td>TRDT max timing case</td>
<td>2707</td>
</tr>
<tr>
<td>739</td>
<td>A-device SRP</td>
<td>2708</td>
</tr>
<tr>
<td>740</td>
<td>B-device SRP</td>
<td>2709</td>
</tr>
<tr>
<td>741</td>
<td>A-device HNP</td>
<td>2710</td>
</tr>
<tr>
<td>742</td>
<td>B-device HNP</td>
<td>2712</td>
</tr>
<tr>
<td>743</td>
<td>HDMI-CEC block diagram</td>
<td>2716</td>
</tr>
<tr>
<td>744</td>
<td>Message structure</td>
<td>2716</td>
</tr>
</tbody>
</table>
### List of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>745</td>
<td>Blocks</td>
<td>2717</td>
</tr>
<tr>
<td>746</td>
<td>Bit timings</td>
<td>2717</td>
</tr>
<tr>
<td>747</td>
<td>Signal free time</td>
<td>2718</td>
</tr>
<tr>
<td>748</td>
<td>Arbitration phase</td>
<td>2718</td>
</tr>
<tr>
<td>749</td>
<td>SFT of three nominal bit periods</td>
<td>2718</td>
</tr>
<tr>
<td>750</td>
<td>Error bit timing</td>
<td>2719</td>
</tr>
<tr>
<td>751</td>
<td>Error handling</td>
<td>2721</td>
</tr>
<tr>
<td>752</td>
<td>TXERR detection</td>
<td>2722</td>
</tr>
<tr>
<td>753</td>
<td>Block diagram of debug infrastructure</td>
<td>2734</td>
</tr>
<tr>
<td>754</td>
<td>Power domains of debug infrastructure</td>
<td>2736</td>
</tr>
<tr>
<td>755</td>
<td>Clock domains of debug infrastructure</td>
<td>2737</td>
</tr>
<tr>
<td>756</td>
<td>SWD successful data transfer</td>
<td>2740</td>
</tr>
<tr>
<td>757</td>
<td>JTAG TAP state machine</td>
<td>2741</td>
</tr>
<tr>
<td>758</td>
<td>Debug and access port connections</td>
<td>2752</td>
</tr>
<tr>
<td>759</td>
<td>APB-D CoreSight component topology</td>
<td>2759</td>
</tr>
<tr>
<td>760</td>
<td>Embedded cross trigger</td>
<td>2766</td>
</tr>
<tr>
<td>761</td>
<td>Mapping of trigger inputs to outputs</td>
<td>2768</td>
</tr>
<tr>
<td>762</td>
<td>ETF state transition diagram</td>
<td>2798</td>
</tr>
<tr>
<td>763</td>
<td>Cortex-M7 CoreSight Topology</td>
<td>2859</td>
</tr>
</tbody>
</table>
1 Documentation conventions

1.1 General information

The STM32H7A3/7B3 and STM32H7B0 devices have an Arm® Cortex®-M7 core.

1.2 List of abbreviations for registers

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

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

---

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

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

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

- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **Double word**: data of 64-bit length.
- **Flash word**: data of 256-bit length
- **IAP (in-application programming)**: IAP is the ability to re-program the Flash memory of a microcontroller while the user program is running.
- **ICP (in-circuit programming)**: ICP is the ability to program the Flash memory of a microcontroller using the JTAG protocol, the SWD protocol or the bootloader while the device is mounted on the user application board.
- **Option bytes**: product configuration bits stored in the Flash memory.
- **AHB**: advanced high-performance bus.
- **AXI**: Advanced extensible interface protocol.
- **PCROP**: proprietary code readout protection.
- **RDP**: readout protection.
- **CD**: CPU power and clock domain.
- **SRD**: Smart Run power and clock domain.

1.4 Availability of peripherals

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

1.5 Availability of security features

For security feature availability please refer to the table below:

<table>
<thead>
<tr>
<th>Security feature</th>
<th>STM32H7B0xB</th>
<th>STM32H7B3xx</th>
<th>STM32H7A3xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded Flash memory (FLASH):</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>– Flash Secure-only area</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security memory management:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>– Secure access mode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>– Root secure services (RSS)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On-the-fly decryption engine (OTFDEC)</td>
<td>Available</td>
<td>Available</td>
<td>Not available</td>
</tr>
<tr>
<td>Cryptographic processor (CRYP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hash processor (HASH)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Memory and bus architecture

## 2.1 System architecture

An AXI bus matrix, two AHB bus matrices and bus bridges allow interconnecting bus masters with bus slaves, as illustrated in Table 2 and Figure 1.

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

<table>
<thead>
<tr>
<th>Bus master / type(1)</th>
<th>Cortex-M7 - AXI</th>
<th>Cortex-M7 - AHBP</th>
<th>Cortex-M7 - ITCM</th>
<th>MDMA - AXI</th>
<th>MDMA - AHB</th>
<th>MDMA - ITM</th>
<th>MDMA - DTM</th>
<th>LTDC</th>
<th>GFX - MMU</th>
<th>DMA1 - MEM.</th>
<th>DMA1 - PERIPH</th>
<th>DMA2 - MEM.</th>
<th>DMA2 - PERIPH</th>
<th>MDMA2 - AHB</th>
<th>OTG HS</th>
<th>BDM1 - AHB</th>
<th>BDM2 - AHB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cortex-M7 - AXI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Cortex-M7 - AHBP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Cortex-M7 - ITCM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MDMA - AXI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MDMA - AHB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MDMA - ITM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MDMA - DTM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LTDC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GFX - MMU</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DMA1 - MEM.</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DMA1 - PERIPH</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DMA2 - MEM.</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DMA2 - PERIPH</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MDMA2 - AHB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OTG HS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BDM1 - AHB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BDM2 - AHB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bus slave / type(1)</th>
<th>Interconnect path and type(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITCM</td>
<td>X</td>
</tr>
<tr>
<td>DTCM</td>
<td>X</td>
</tr>
<tr>
<td>Flash memory(5)</td>
<td>X</td>
</tr>
<tr>
<td>AXI SRAM1</td>
<td>X</td>
</tr>
<tr>
<td>AXI SRAM2</td>
<td>X</td>
</tr>
<tr>
<td>AXI SRAM3</td>
<td>X</td>
</tr>
<tr>
<td>GFX-MMU</td>
<td>X</td>
</tr>
<tr>
<td>OCTOSPI1</td>
<td>X</td>
</tr>
<tr>
<td>OCTOSPI2</td>
<td>X</td>
</tr>
<tr>
<td>OTFDEC1</td>
<td>X</td>
</tr>
<tr>
<td>OTFDEC2(4)</td>
<td>X</td>
</tr>
<tr>
<td>FMC</td>
<td>X</td>
</tr>
<tr>
<td>AHB SRAM1</td>
<td>X</td>
</tr>
<tr>
<td>AHB SRAM2</td>
<td>X</td>
</tr>
<tr>
<td>AHB1 peripherals</td>
<td>X</td>
</tr>
<tr>
<td>APB1 peripherals</td>
<td>X</td>
</tr>
<tr>
<td>APB2 peripherals</td>
<td>X</td>
</tr>
<tr>
<td>AHB2 peripherals</td>
<td>X</td>
</tr>
<tr>
<td>AHB3 peripherals</td>
<td>X</td>
</tr>
<tr>
<td>APB3 peripherals</td>
<td>X</td>
</tr>
<tr>
<td>AHB4 peripherals</td>
<td>X</td>
</tr>
<tr>
<td>APB4 peripherals</td>
<td>X</td>
</tr>
</tbody>
</table>

(1) Bus master / type: Cortex-M7 - AXI, Cortex-M7 - AHBP, Cortex-M7 - ITCM, MDMA - AXI, MDMA - AHB, MDMA - ITM, MDMA - DTM, LTDC, GFX - MMU, DMA1 - MEM, DMA1 - PERIPH, DMA2 - MEM, DMA2 - PERIPH, MDMA2 - AHB, OTG HS, BDM1 - AHB, BDM2 - AHB.

(2) Interconnect path and type: X.
### Table 2. Bus-master-to-bus-slave interconnect (continued)

<table>
<thead>
<tr>
<th>Bus slave / type(1)</th>
<th>Interconnect path and type(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRD SRAM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X    -                      -  -  X - - -  -   X  X  X  X -   -  -  X</td>
</tr>
<tr>
<td>Backup RAM</td>
<td>X    -                      -  -  X - - -  -   X  X  X  X -   -  -  X</td>
</tr>
</tbody>
</table>

1. **Bold** font type denotes 64-bit bus, plain type denotes 32-bit bus.
2. Table cells indicate access and usage, path and type:
   - "X" = access possible, "-" = access not possible, shading = access useful/usable.
3. Bank 1 is limited to 128 Kbytes on STM32H7B3/B0 devices.
4. OTFDEC2 is available only on STM32H7B0 and STM32H7B3 devices.
1. Bank 1 is limited to 128 Kbytes on STM32H7B0 devices. STM32H7A3xG and STM32H7A3xI/7B3xx devices feature two banks of 512 Kbytes and 1 Mbyte each, respectively.

2. OTFDEC1 and OTFDEC2 are available only on STM32H7B0 and STM32 H7B3 devices.
2.1.1 Bus matrices

AXI bus matrix in CD domain

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

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

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

AHB bus matrices in CD domain

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

The arbitration uses a round-robin algorithm.

2.1.2 TCM buses

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

2.1.3 Bus-to-bus bridges

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

The AHB/APB bridges allow connecting peripherals on APB1 and APB2 to AHB1. Other bridges in the CD domain and in the SRD domain allow connecting APB3 to AHB3 and APB4 to AHB4, respectively. These AHB/APB bridges provide full synchronous interfacing, which allows the APB peripherals to operate with clocks independent of AHB that they connect to.

The AHB/APB bridges also allow APB1 and APB2 peripherals to connect to DMA1 and DMA2 peripheral buses, respectively, without transiting through AHB1.

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

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

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

AXI to CD AHB
A 32-bit bus connects the AXI bus matrix to the AHB bus matrix. It allows AXI bus master to access resources (bus slaves) inside the CD domain.

CD AHB to AXI bus
A 32-bit bus connects the AHB bus matrix to the AXI bus matrix. It allows AHB bus masters to access resources (bus slaves) inside the CD domain and indirectly via the CD-to-SRD AHB in the SRD domain.

AXI to SRD AHB
This 32-bit bus connects the CD domain to the SRD domain AHB bus matrix. It allows bus masters in the CD domain to access resources (bus slaves) in the SRD domain.

CD AHB to SRD AHB
This 32-bit bus connects the CD domain to the SRD domain AHB bus matrix. It allows bus masters in the CD domain to access resources (bus slaves) in the SRD domain.

2.1.5 CPU buses

Cortex®-M7 AXIM bus
The Cortex®-M7 CPU uses the 64-bit AXIM bus to access all memories (excluding ITCM, and DTCM) and AHB3, AHB4, APB3 and APB4 peripherals (excluding AHB1, APB1 and APB2 peripherals).

The AXIM bus connects the CPU to the AXI bus matrix in the CD domain.

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

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

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

Cortex®-M7 AHBP bus
The Cortex®-M7 CPU uses the 32-bit AHBP bus for accessing AHB1, AHB2, APB1 and APB2 peripherals via the AHB bus matrix in the CD domain.
2.1.6 Bus master peripherals

SDMMC1
The SDMMC1 uses a 32-bit bus, connected to the AXI bus matrix, through which it can access internal AXI SRAM and Flash memories, and external memories through the OCTOSPI1/2 controllers and the FMC.

SDMMC2
The SDMMC2 uses a 32-bit bus, connected to the AHB bus matrix in CD domain. Through the system bus matrices, it can access the internal AXI SRAM1, AXI SRAM2, AXI SRAM3, AHB SRAM1, AHB SRAM2 and Flash memories, and external memories through the OCTOSPI1/2 controllers and the FMC.

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

The MDMA is optimized for DMA data transfers between memories since it supports linked list transfers that allow performing a chained list of transfers without the need for CPU intervention. Through the system bus matrices and the Cortex-M7 AHBS slave bus, the MDMA can access all internal and external memories through the OCTOSPI1/2 controllers and the FMC.

DMA1 and DMA2 controllers
The DMA1 and DMA2 controllers have two 32-bit buses - memory bus and peripheral bus, connected to the AHB bus matrix in CD domain.

The memory bus allows DMA data transfers between memories. Through the system bus matrices, the memory bus can access external memories through the OCTOSPI1/2 controllers and the FMC and all internal memories except ITCM and DTCM.

The peripheral bus allows DMA data transfers between two peripherals, between two memories or between a peripheral and a memory. Through the system bus matrices, the peripheral bus can access external memories through the OCTOSPI controller and the FMC, all AHB and APB peripherals and all internal memories except ITCM and DTCM. A direct access to APB1 and APB2 is available, without passing through AHB1. Direct path to APB1 and APB2 bridges allows reducing the bandwidth usage on AHB1 bus by improving data treatment efficiency for APB and AHB peripherals.

BDMA1 controller
The BDMA1 controller is connected to the AHB bus matrix in the CD domain. It is dedicated to the DFSDM1 which is connected to the APB2 bus. The BDMA1 allows DMA data transfers between the DFSDM and AHB SRAM or AHB1, APB1 and APB2 peripherals.

BDMA2 controller
The BDMA2 controller uses a 32-bit bus, connected to the AHB bus matrix in SRD domain, for DMA data transfers between two peripherals, between two memories or between a peripheral and a memory. BDMA2 transfers are limited to the SRD domain resources. It can access the internal SRD SRAM, backup SRAM, and AHB4 and APB4 peripherals through the AHB bus matrix in the SRD domain.
Chrom-Art Accelerator (DMA2D)

The DMA2D graphics accelerator uses a 64-bit bus, connected to the AXI bus matrix. Through the system bus matrices, internal AXI SRAM1, AXI SRAM2, AXI SRAM3, AHB SRAM1, AHB SRAM2 and Flash memories, and external memories through the OCTOSPI1/2 controllers and the FMC.

LCD-TFT controller (LTDC)

The LCD-TFT display controller, LTDC, uses a 64-bit bus, connected to the AXI bus matrix, through which it can access internal AXI SRAM1, AXI SRAM2, AXI SRAM3 and Flash memories, and external memories through the OCTOSPI1/2 controllers and the FMC.

OTG_HS peripheral

The OTG_HS peripheral uses a 32-bit bus, connected to the AHB bus matrix in the CD domain. Through the system bus matrix, it can access all internal memories except ITCM and DTCM, and external memories through the OCTOSPI1/2 controllers and the FMC.

2.1.7 Clocks to functional blocks

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

2.2 AXI interconnect matrix (AXIM)

2.2.1 AXI introduction

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

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

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

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

2.2.2 AXI interconnect main features

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

Block diagram

The AXI interconnect is shown in Figure 2.

Figure 2. AXI interconnect
ASIB configuration

Table 3 summarizes the characteristics of the ASIBs.

Table 3. ASIB configuration

<table>
<thead>
<tr>
<th>ASIB</th>
<th>Connected master</th>
<th>Protocol</th>
<th>Bus width</th>
<th>R/W issuing</th>
</tr>
</thead>
<tbody>
<tr>
<td>INI 1</td>
<td>AHB from CD domain</td>
<td>AHB-lite</td>
<td>32</td>
<td>1/4</td>
</tr>
<tr>
<td>INI 2</td>
<td>Cortex-M7</td>
<td>AXI4</td>
<td>64</td>
<td>7/32</td>
</tr>
<tr>
<td>INI 3</td>
<td>SDMMC1</td>
<td>AHB-lite</td>
<td>32</td>
<td>1/4</td>
</tr>
<tr>
<td>INI 4</td>
<td>MDMA</td>
<td>AXI4</td>
<td>64</td>
<td>4/1</td>
</tr>
<tr>
<td>INI 5</td>
<td>DMA2D</td>
<td>AXI4</td>
<td>64</td>
<td>2/1</td>
</tr>
<tr>
<td>INI 6</td>
<td>LTDC</td>
<td>AXI4</td>
<td>64</td>
<td>2/1</td>
</tr>
<tr>
<td>INI 7</td>
<td>GFXMMU</td>
<td>AHB-lite</td>
<td>32</td>
<td>1/1</td>
</tr>
</tbody>
</table>

AMIB configuration

Table 4 summarizes the characteristics of the AMIBs.

Table 4. AMIB configuration

<table>
<thead>
<tr>
<th>AMIB</th>
<th>Connected slave</th>
<th>Protocol</th>
<th>Bus width</th>
<th>R/W/Total acceptance</th>
</tr>
</thead>
<tbody>
<tr>
<td>TARG 1</td>
<td>Peripheral 3, SRD AHB and user OTP</td>
<td>AXI4(^{(1)})</td>
<td>32</td>
<td>1/1/1</td>
</tr>
<tr>
<td>TARG 2</td>
<td>CD AHB</td>
<td>AXI4(^{(1)})</td>
<td>32</td>
<td>1/1/1</td>
</tr>
<tr>
<td>TARG 3</td>
<td>Flash A and Flash B</td>
<td>AXI4</td>
<td>64</td>
<td>3/4/7</td>
</tr>
<tr>
<td>TARG 4</td>
<td>OCTOSPI2</td>
<td>AXI4</td>
<td>64</td>
<td>3/2/5</td>
</tr>
<tr>
<td>TARG 5</td>
<td>FMC</td>
<td>AXI4</td>
<td>64</td>
<td>3/3/6</td>
</tr>
<tr>
<td>TARG 6</td>
<td>OCTOSPI1</td>
<td>AXI4</td>
<td>64</td>
<td>2/1/3</td>
</tr>
<tr>
<td>TARG 7</td>
<td>AXI SRAM1</td>
<td>AXI3</td>
<td>64</td>
<td>2/2/4</td>
</tr>
<tr>
<td>TARG 8</td>
<td>AXI SRAM2</td>
<td>AXI3</td>
<td>64</td>
<td>2/2/4</td>
</tr>
<tr>
<td>TARG 9</td>
<td>AXI SRAM3</td>
<td>AXI3</td>
<td>64</td>
<td>2/2/4</td>
</tr>
<tr>
<td>TARG 10</td>
<td>GFXMMU(^{(1)})</td>
<td>AXI4</td>
<td>32</td>
<td>2/2/4</td>
</tr>
</tbody>
</table>

1. Conversion to AHB protocol is done via an AXI/AHB bridge sitting between AXI interconnect and the connected slave.

Quality of service (QoS)

The AXI switch matrix uses a priority-based arbitration when two ASIB simultaneously attempt to access the same AMIB. Each ASIB has programmable read channel and write channel priorities, known as QoS, from 0 to 15, such that the higher the value, the higher the priority. The read channel QoS value is programmed in the AXI interconnect - INI x read QoS register (AXI_INIX_READ_QOS), and the write channel in the AXI interconnect - INI x write QoS register (AXI_INIX_WRITE_QOS). The default QoS value for all channels is 0 (lowest priority).
If two coincident transactions arrive at the same AMIB, the higher priority transaction passes before the lower priority. If the two transactions have the same QoS value, then a least-recently-used (LRU) priority scheme is adopted.

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

**Global programmer view (GPV)**

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

### 2.2.4 AXI interconnect registers

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

Address offset: 0x1FD0  
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4KCOUNT[3:0]</td>
<td>JEP106CON[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

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

Address offset: 0x1FE0  
Reset value: 0x0000 0000

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

108/2974  RM0455 Rev 3
**AXI interconnect - peripheral ID1 register (AXI_PERIPH_ID_1)**

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

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

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

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

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

Address offset: 0x1FE8
Reset value: 0x0000 002B

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

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

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

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

**Notes:**
- **REVISION[3:0]**: Peripheral revision number
  - 0x2: r0p2
- Bit 3 **JEDEC**: JEP106 code flag
  - 0x1: JEDEC allocated code
- Bits 2:0 **JEP106ID[6:4]**: JEP106 Identity bits 4 to 6
  - 0x3: Arm® JEDEC code
### AXI interconnect - peripheral ID3 register (AXI_PERIPH_ID_3)

Address offset: 0x1FEC  
Reset value: 0x0000 0000

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

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

<table>
<thead>
<tr>
<th>REV_AND[3:0]</th>
<th>CUST_MOD_NUM[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

- Bits 31:8: Reserved, must be kept at reset value.
- Bits 7:4: REV_AND[3:0]: Customer version  
  0: None
- Bits 3:0: CUST_MOD_NUM[3:0]: Customer modification  
  0: None

### AXI interconnect - component ID0 register (AXI_COMP_ID_0)

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

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

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

<table>
<thead>
<tr>
<th>PREAMBLE[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
</tr>
</tbody>
</table>

- Bits 31:8: Reserved, must be kept at reset value.
- Bits 7:0: PREAMBLE[7:0]: Preamble bits 0 to 7  
  0xD: Common ID value
### AXI interconnect - component ID1 register (AXI_COMP_ID_1)

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

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

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

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

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

### AXI interconnect - component ID2 register (AXI_COMP_ID_2)

Address offset: 0x1FF8
Reset value: 0x0000 0005

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

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

Bits 7:0 **PREAMBLE[19:12]**: Preamble bits 12 to 19
- 0x05: Common ID value
AXI interconnect - component ID3 register (AXI_COMP_ID_3)

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

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

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

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

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

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

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

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

Bit 1 WRITE_ISS_OVERRIDE: Switch matrix write issuing override for target
0: Default issuing capability
1: Set switch matrix write issuing capability to 1 (see Table 4: AMIB configuration)

Bit 0 READ_ISS_OVERRIDE: Switch matrix read issuing override for target
0: Default issuing capability
1: Set switch matrix read issuing capability to 1 (see Table 4: AMIB configuration)
AXI interconnect - TARG x bus matrix functionality 2 register  
(AXI_TARGx_FN_MOD2)

Address offset: 0x1024 + 0x1000 * x, (x = 1, 2, 7 to 10)
Reset value: 0x0000 0000

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

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

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

0: Normal operation
1: Disable packing

---

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

Address offset: 0x102C + 0x1000 * x, (x = 1, 2, 10)
Reset value: 0x0000 0000

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

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

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

0: Long bursts can not be generated at the output of the ASIB
1: Long bursts can be generated at the output of the ASIB
AXI interconnect - TARG x issuing functionality modification register  
(AXI_TARGx_FN_MOD)

Address offset: 0x1108 + 0x1000 * x, (x = 1, 2, 7 to 10)  
Reset value: 0x0000 0000

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

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

Bit 1 WRITE_ISS_OVERRIDE: Override AMIB write issuing capability  
0: Default issuing capability  
1: Force issuing capability to 1 (see Table 4: AMIB configuration)

Bit 0 READ_ISS_OVERRIDE: Override AMIB read issuing capability  
0: Default issuing capability  
1: Force issuing capability to 1 (see Table 4: AMIB configuration)

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

Address offset: 0x41024 + 0x1000 * x, (x = 1, 3, 7)  
Reset value: 0x0000 0000

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

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

Bit 0 BYPASS_MERGE: Disables alteration of transactions by the up-sizer unless required by the protocol  
0: Normal operation  
1: Transactions pass through unaltered where allowed
AXI interconnect - INI x AHB functionality modification register
(AXI_INIx_FN_MOD_AHB)

Address offset: 0x41028 + 0x1000 * x, (x = 1, 3, 7)
Reset value: 0x0000 0000

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

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

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

Bit 1 **WR_INC_OVERRIDE**: Converts all AHB-Lite read transactions to a series of single beat AXI transactions.

  0: Override disabled
  1: Override enabled

Bit 0 **RD_INC_OVERRIDE**: Converts all AHB-Lite write transactions to a series of single beat AXI transactions, and each AHB-Lite write beat is acknowledged with the AXI buffered write response.

  0: Override disabled
  1: Override enabled

AXI interconnect - INI x read QoS register (AXI_INIx_READ_QOS)

Address offset: 0x41100 + 0x1000 * x, (x = 1 to 7)
Reset value: 0x0000 0000

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

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

<table>
<thead>
<tr>
<th></th>
<th>WR_INC_OVERRIDE</th>
<th>RD_INC_OVERRIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 3:0 **AR_QOS[3:0]**: Read channel QoS setting

  0x0: Lowest priority
  0xF: Highest priority
### AXI interconnect - INI x write QoS register (AXI_INIx_WRITE_QOS)

Address offset: 0x41104 + 0x1000 * x, (x = 1 to 7)

Reset value: 0x0000 0000

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

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

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

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

Address offset: 0x41108 + 0x1000 * x, (x = 1 to 7)

Reset value: 0x0000 0000

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

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

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

Bit 0  **READ_ISS_OVERRIDE**: Override ASIB read issuing capability
- 0: Normal issuing capability
- 1: Force issuing capability to 1
### AXI interconnect register map

Table 5. AXI interconnect register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1FD0</td>
<td>AXI_PERIPH_ID_4</td>
<td>0x1FD8</td>
<td>AXI_PERIPH_ID_6</td>
<td>0x1FDC</td>
<td>AXI_PERIPH_ID_7</td>
<td>0x1FE0</td>
<td>AXI_PERIPH_ID_0</td>
<td>0x1FE4</td>
<td>AXI_PERIPH_ID_1</td>
<td>0x1FE8</td>
<td>AXI_PERIPH_ID_2</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td>JEP106ID[6:4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00000100</td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>10110100</td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td>0x1FD4</td>
<td>0x1FD8</td>
<td>AXI_PERIPH_ID_6</td>
<td>0x1FD8</td>
<td>AXI_PERIPH_ID_6</td>
<td>0x1FD8</td>
<td>AXI_PERIPH_ID_6</td>
<td>0x1FD8</td>
<td>AXI_PERIPH_ID_6</td>
<td>0x1FE0</td>
<td>AXI_PERIPH_ID_0</td>
</tr>
<tr>
<td></td>
<td>AXI_PERIPH_ID_5</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>PARTNUM[7:0]</td>
<td></td>
<td>JEP106ID[3:0]</td>
<td></td>
<td>REV_AND[3:0]</td>
</tr>
<tr>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td>JEP106ID[6:4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1FD4</td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>10110100</td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td>AXI_PERIPH_ID_5</td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1FD8</td>
<td>0x1FD8</td>
<td>AXI_PERIPH_ID_6</td>
<td>0x1FD8</td>
<td>AXI_PERIPH_ID_6</td>
<td>0x1FD8</td>
<td>AXI_PERIPH_ID_6</td>
<td>0x1FD8</td>
<td>AXI_PERIPH_ID_6</td>
<td>0x1FE0</td>
<td>AXI_PERIPH_ID_0</td>
</tr>
<tr>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td>JEP106ID[6:4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1FD8</td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>10110100</td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td>AXI_PERIPH_ID_6</td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>10110100</td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td>AXI_PERIPH_ID_7</td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>10110100</td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>10110100</td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>0x1FE4</td>
<td>AXI_PERIPH_ID_1</td>
<td></td>
<td>JEP106ID[3:0]</td>
<td></td>
<td>JEP106ID[3:0]</td>
<td></td>
<td>PARTNUM[11:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10110100</td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1FE8</td>
<td>AXI_PERIPH_ID_2</td>
<td></td>
<td>REVISION[3:0]</td>
<td></td>
<td>JEP106ID[3:0]</td>
<td></td>
<td>PARTNUM[11:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1FF0</td>
<td>AXI_COMP_ID_0</td>
<td></td>
<td>PREAMBLE[7:0]</td>
<td></td>
<td>PREAMBLE[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1FF4</td>
<td>AXI_COMP_ID_1</td>
<td></td>
<td>CLASS[3:0]</td>
<td></td>
<td>PREAMBLE[11:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11111100</td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 5. AXI interconnect register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset Register</th>
<th>Reset value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1FF8</td>
<td>AXI_COMP_ID_2</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>0 0 0 0 0 1 0 1</td>
<td>PREAMBLE[19:12]</td>
</tr>
<tr>
<td>0x1FFC</td>
<td>AXI_COMP_ID_3</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>1 0 1 1 0 0 0 1</td>
<td>PREAMBLE[27:20]</td>
</tr>
<tr>
<td>0x2000 - 0x2004</td>
<td>Reserved</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0x2008</td>
<td>AXI_TARG1_FN_MOD_ISS_BM</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td>WRITE_ISS_OVERRIDE</td>
</tr>
<tr>
<td>0x200C - 0x2020</td>
<td>Reserved</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td>READ_ISS_OVERRIDE</td>
</tr>
<tr>
<td>0x2024</td>
<td>AXI_TARG1_FN_MOD2</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td>BYPASS_MERGE</td>
</tr>
<tr>
<td>0x2028</td>
<td>Reserved</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0x202C</td>
<td>AXI_TARG1_FN_MOD_LB</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td>FN_MOD_LB</td>
</tr>
<tr>
<td>0x2030 - 0x2104</td>
<td>Reserved</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td>WRITE_ISS_OVERRIDE</td>
</tr>
<tr>
<td>0x2108</td>
<td>AXI_TARG1_FN_MOD</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td>READ_ISS_OVERRIDE</td>
</tr>
<tr>
<td>0x210C - 0x3004</td>
<td>Reserved</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>
## Table 5. AXI interconnect register map and reset values (continued)

| Offset     | Register name                      | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|------------|-----------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x3008     | AXI_TARG2_FN_MOD_ISS_BM           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|            | Reset value                       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x300C - 0x3020 | Reserved                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x3024     | AXI_TARG2_FN_MOD2                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|            | Reset value                       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x302B     | Reserved                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x302C     | AXI_TARG2_FN_MOD_LB              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|            | Reset value                       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x3030 - 0x3104 | Reserved                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x3108     | AXI_TARG2_FN_MOD                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|            | Reset value                       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x310C - 0x4004 | Reserved                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4008     | AXI_TARG3_FN_MOD_ISS_BM          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|            | Reset value                       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x400C - 0x5004 | Reserved                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
### Table 5. AXI interconnect register map and reset values (continued)

| Offset  | Register name         | Bit 31 | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9  | Bit 8  | Bit 7  | Bit 6  | Bit 5  | Bit 4  | Bit 3  | Bit 2  | Bit 1  | Bit 0  |
|---------|-----------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0x5008  | AXI_TARG4_            |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | FN_MOD_               |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | ISS_BM                |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | Reset value           |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x500C  | Reserved              |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x6008  | AXI_TARG5_            |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | FN_MOD_               |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | ISS_BM                |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | Reset value           |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x600C  | Reserved              |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x7008  | AXI_TARG6_            |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | FN_MOD_               |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | ISS_BM                |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | Reset value           |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x700C  | Reserved              |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x8008  | AXI_TARG7_            |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | FN_MOD_               |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | ISS_BM                |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | Reset value           |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x800C  | Reserved              |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x8024  | AXI_TARG7_            |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | FN_MOD2               |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|         | Reset value           |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| Offset       | Register name                                      | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------------|----------------------------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x8028 - 0x8104 | Reserved                                           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x8108       | AXI_TARG7_FN_MOD                                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value                                        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x810C - 0x9004 | Reserved                                           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x9008       | AXI_TARG8_FN_MOD_11ISS_BM                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value                                        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x900C - 0x9020 | Reserved                                           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x9024       | AXI_TARG8_FN_MOD2                                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value                                        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x9028 - 0x9104 | Reserved                                           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x9108       | AXI_TARG8_FN_MOD                                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value                                        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x910C - 0xA004 | Reserved                                           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Table 5. AXI interconnect register map and reset values (continued)

| Offset   | Register name                  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|--------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xA008   | AXI_TARG9_                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | FN_MOD_                         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | ISS_BM                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xA00C-0xA20 | Reserved                        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xA024   | AXI_TARG9_                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | FN_MOD2                         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xA028-0xA104 | Reserved                       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xA108   | AXI_TARG9_                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | FN_MOD                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xA10C-0xB004 | Reserved                       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xB008   | AXI_TARG10_                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | FN_MOD_                         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | ISS_BM                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xB00C-0xB020 | Reserved                       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xB024   | AXI_TARG10_                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | FN_MOD2                         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
### Table 5. AXI interconnect register map and reset values (continued)

| Offset       | Register name            | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------------|--------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xB028-0xB104| Reserved                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xB108       | AXI_TARG10_           FN_MOD |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xB10C-0xC004| Reserved                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xC02C       | AXI_TARG10_           FN_MOD_LB |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xC030-C0104| Reserved                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x42024      | AXI_INI1_           FN_MOD2 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x42028      | AXI_INI1_FN_MOD_AHB     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4202C-0x420FC| Reserved             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x42100      | AXI_INI1_READ_QOS      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x42104      | AXI_INI1_WRITE_QOS     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
### Table 5. AXI interconnect register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>AR_QOS [3:0]</th>
<th>WR, INC. OVERRIDE</th>
<th>RD, INC. OVERRIDE</th>
<th>BYPASS, MERGE</th>
<th>WRITE ISS OVERRIDE</th>
<th>READ ISS OVERRIDE</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x42108</td>
<td>AXI_INI1_FN_MOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x4210C-0x430FC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x43100</td>
<td>AXI_INI2_READ_QOS</td>
<td>AR_QOS [3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x43104</td>
<td>AXI_INI2_WRITE_QOS</td>
<td>AW_QOS [3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x43108</td>
<td>AXI_INI2_FN_MOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x4310C-0x44020</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44024</td>
<td>AXI_INI3_FN_MOD2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44028</td>
<td>AXI_INI3_FN_MOD_AHB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x4402C-0x440FC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44100</td>
<td>AXI_INI3_READ_QOS</td>
<td>AR_QOS [3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>0x44104</td>
<td>AXI_INI3_WRITE_QOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44108</td>
<td>AXI_INI3_FN_MOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4410C-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x450FC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x45100</td>
<td>AXI_INI4_READ_QOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x45104</td>
<td>AXI_INI4_WRITE_QOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x45108</td>
<td>AXI_INI4_FN_MOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4510C-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x460FC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x46100</td>
<td>AXI_INI5_READ_QOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x46104</td>
<td>AXI_INI5_WRITE_QOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x46108</td>
<td>AXI_INI5_FN_MOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4610C-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x470FC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x47100</td>
<td>AXI_INI6_READ_QOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x47104</td>
<td>AXI_INI6_WRITE_QOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x47108</td>
<td>AXI_INI6_FN_MOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4710C-0x48020</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x48024</td>
<td>AXI_INI7_FN_MOD2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x48028</td>
<td>AXI_INI7_FN_MOD_AHB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4802C-0x480FC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x48100</td>
<td>AXI_INI7_READ_QOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x48104</td>
<td>AXI_INI7_WRITE_QOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Refer to Section 2.3 on page 128 for the register boundary addresses.

| Offset  | Register name         | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0   |
|---------|-----------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x48108 | AXI_INIT_             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | FN_MOD               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | WRITE_ISS_OVERRIDE   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | READ_ISS_OVERRIDE    | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
2.3 Memory organization

2.3.1 Introduction

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

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

The addressable memory space is divided into eight main blocks, of 512 Mbytes each.
2.3.2 Register boundary addresses

All the memory map areas that are not allocated to on-chip memories and peripherals are considered “Reserved”. For the detailed mapping of available memory areas, refer to Section 2.4: Embedded SRAM and Section 2.5: Flash memory overview.

The following table gives the register boundary addresses of the peripherals available in the devices.

<table>
<thead>
<tr>
<th>Boundary address</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Register map</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x58025800 - 0x58025BFF</td>
<td>DMAMUX2</td>
<td>AHB4 (SRD)</td>
<td>Section 17.6: DMAMUX registers</td>
</tr>
<tr>
<td>0x58025400 - 0x580257FF</td>
<td>BDMA2</td>
<td></td>
<td>Section 16.6: BDMA registers</td>
</tr>
<tr>
<td>0x58024800 - 0x58024BFF</td>
<td>PWR</td>
<td></td>
<td>Section 6.8: PWR registers</td>
</tr>
<tr>
<td>0x58024400 - 0x580247FF</td>
<td>RCC</td>
<td></td>
<td>Section 8.7: RCC registers</td>
</tr>
<tr>
<td>0x58022800 - 0x58022BFF</td>
<td>GPIOK</td>
<td></td>
<td>Section 11.4: GPIO registers</td>
</tr>
<tr>
<td>0x58022400 - 0x580227FF</td>
<td>GPIOJ</td>
<td></td>
<td>Section 11.4: GPIO registers</td>
</tr>
<tr>
<td>0x58022000 - 0x580223FF</td>
<td>GPIOI</td>
<td></td>
<td>Section 11.4: GPIO registers</td>
</tr>
<tr>
<td>0x58021C00 - 0x58021FFF</td>
<td>GPIOH</td>
<td></td>
<td>Section 11.4: GPIO registers</td>
</tr>
<tr>
<td>0x58021800 - 0x58021BFF</td>
<td>GPIOG</td>
<td></td>
<td>Section 11.4: GPIO registers</td>
</tr>
<tr>
<td>0x58021400 - 0x580217FF</td>
<td>GPIOF</td>
<td></td>
<td>Section 11.4: GPIO registers</td>
</tr>
<tr>
<td>0x58021000 - 0x580213FF</td>
<td>GPIOE</td>
<td></td>
<td>Section 11.4: GPIO registers</td>
</tr>
<tr>
<td>0x58020C00 - 0x58020FFF</td>
<td>GPIOD</td>
<td></td>
<td>Section 11.4: GPIO registers</td>
</tr>
<tr>
<td>0x58020800 - 0x58020BFF</td>
<td>GPIOC</td>
<td></td>
<td>Section 11.4: GPIO registers</td>
</tr>
<tr>
<td>0x58020400 - 0x580207FF</td>
<td>GPIOB</td>
<td></td>
<td>Section 11.4: GPIO registers</td>
</tr>
<tr>
<td>0x58020000 - 0x580203FF</td>
<td>GPIOA</td>
<td></td>
<td>Section 11.4: GPIO registers</td>
</tr>
</tbody>
</table>
### Table 6. Register boundary addresses (1)

<table>
<thead>
<tr>
<th>Boundary address</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Register map</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x58006C00 - 0x580073FF</td>
<td>DFSDM2 (1 filter)</td>
<td>APB4 (SRD)</td>
<td>Section 33.7: DFSDM channel y registers (y=0..7) and Section 33.8: DFSDM filter x module registers (x=0..7)</td>
</tr>
<tr>
<td>0x58006800 - 0x58006BFF</td>
<td>DTS</td>
<td></td>
<td>Section 28.6: DTS registers</td>
</tr>
<tr>
<td>0x58004800 - 0x58004BFF</td>
<td>IWDG</td>
<td></td>
<td>Section 49.4: IWDG registers</td>
</tr>
<tr>
<td>0x58004400 - 0x580047FF</td>
<td>Tamper and backup registers</td>
<td></td>
<td>Section 51.6: TAMP registers</td>
</tr>
<tr>
<td>0x58004000 - 0x580043FF</td>
<td>RTC</td>
<td></td>
<td>Section 50.6: RTC registers</td>
</tr>
<tr>
<td>0x58003C00 - 0x58003FFF</td>
<td>VREF</td>
<td></td>
<td>Section 30.3: VREFBUF registers</td>
</tr>
<tr>
<td>0x58003800 - 0x58003BFF</td>
<td>COMP1 - COMP2</td>
<td></td>
<td>Section 31.6: COMP registers</td>
</tr>
<tr>
<td>0x58003400 - 0x580037FF</td>
<td>DAC2</td>
<td></td>
<td>Section 29.7: DAC registers</td>
</tr>
<tr>
<td>0x58002800 - 0x58002BFF</td>
<td>LPTIM3</td>
<td></td>
<td>Section 47.7: LPTIM registers</td>
</tr>
<tr>
<td>0x58002400 - 0x580027FF</td>
<td>LPTIM2</td>
<td></td>
<td>Section 47.7: LPTIM registers</td>
</tr>
<tr>
<td>0x58001C00 - 0x58001FFF</td>
<td>I2C4</td>
<td></td>
<td>Section 52.7: I2C registers</td>
</tr>
<tr>
<td>0x58001400 - 0x580017FF</td>
<td>SPI/I2S6</td>
<td></td>
<td>Section 55.11: SPI/I2S registers</td>
</tr>
<tr>
<td>0x58000C00 - 0x58000FFF</td>
<td>LPUART1</td>
<td></td>
<td>Section 54.6: LPUART registers</td>
</tr>
<tr>
<td>0x58000400 - 0x580007FF</td>
<td>SYSCFG</td>
<td></td>
<td>Section 12.4: SYSCFG registers</td>
</tr>
<tr>
<td>0x58000000 - 0x580003FF</td>
<td>EXTI</td>
<td></td>
<td>Section 20.6: EXTI registers</td>
</tr>
<tr>
<td>0x5200C000 - 0x5200EFFF</td>
<td>GFXMMU</td>
<td>AHB3 (CD)</td>
<td>Section 21.5: Graphic MMU registers</td>
</tr>
<tr>
<td>0x5200BC00 - 0x5200BFFF</td>
<td>OTFDEC2</td>
<td></td>
<td>Section 41.6: OTFDEC registers</td>
</tr>
<tr>
<td>0x5200B800 - 0x5200BBFF</td>
<td>OTFDEC1</td>
<td></td>
<td>Section 41.6: OTFDEC registers</td>
</tr>
<tr>
<td>0x5200B400 - 0x5200B7FF</td>
<td>OCTOSPI I/O manager</td>
<td></td>
<td>Section 25.4: OCTOSPI I/O manager registers</td>
</tr>
<tr>
<td>0x5200B000 - 0x5200B3FF</td>
<td>Delay Block OCTOSPI2</td>
<td></td>
<td>Section 26.4: DLYB registers</td>
</tr>
<tr>
<td>0x5200A000 - 0x5200AFFF</td>
<td>OCTOSPI2</td>
<td></td>
<td>Section 24.5: OCTOSPI registers</td>
</tr>
<tr>
<td>0x52009000 - 0x520093FF</td>
<td>RAMECC</td>
<td></td>
<td>Section 3.4: RAMECC registers</td>
</tr>
<tr>
<td>Boundary address</td>
<td>Peripheral</td>
<td>Bus</td>
<td>Register map</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x520008000 - 0x5200083FF</td>
<td>Delay Block SDMMC1</td>
<td></td>
<td>Section 26.4: DLYB registers</td>
</tr>
<tr>
<td>0x520070000 - 0x52007FFF</td>
<td>SDMMC1</td>
<td></td>
<td>Section 60.9: SDMMC registers</td>
</tr>
<tr>
<td>0x520060000 - 0x520063FF</td>
<td>Delay Block OCTOSPI1</td>
<td></td>
<td>Section 26.4: DLYB registers</td>
</tr>
<tr>
<td>0x520050000 - 0x52005FFF</td>
<td>OCTOSPI1 control registers</td>
<td></td>
<td>Section 24.5: OCTOSPI registers</td>
</tr>
<tr>
<td>0x520040000 - 0x52004FFF</td>
<td>FMC control registers</td>
<td>AHB3 (CD)</td>
<td>Section 23.7.6: NOR/PSRAM controller registers, Section 23.8.7: NAND Flash controller registers, Section 23.9.5: SDRAM controller registers</td>
</tr>
<tr>
<td>0x520030000 - 0x52003FFF</td>
<td>JPEG</td>
<td></td>
<td>Section 37.5: JPEG codec registers</td>
</tr>
<tr>
<td>0x520020000 - 0x52002FFF</td>
<td>Flash interface registers</td>
<td></td>
<td>Section 4.9: FLASH registers</td>
</tr>
<tr>
<td>0x520010000 - 0x520011FFF</td>
<td>Chrom-Art (DMA2D)</td>
<td></td>
<td>Section 18.6: DMA2D registers</td>
</tr>
<tr>
<td>0x520000000 - 0x520001FFF</td>
<td>MDMA</td>
<td></td>
<td>Section 14.5: MDMA registers</td>
</tr>
<tr>
<td>0x510000000 - 0x510003FFF</td>
<td>GPV</td>
<td></td>
<td>Section 2.2.4: AXI interconnect registers</td>
</tr>
<tr>
<td>0x500030000 - 0x50003FFF</td>
<td>WWDG</td>
<td></td>
<td>Section 48.4: WWDG interrupts</td>
</tr>
<tr>
<td>0x500010000 - 0x500011FFF</td>
<td>LTDC</td>
<td></td>
<td>Section 36.7: LTDC registers</td>
</tr>
<tr>
<td>0x48022C000 - 0x48022FFFF</td>
<td>BDMA1 (dedicated to DFSDM)</td>
<td></td>
<td>Section 16.6: BDMA registers</td>
</tr>
<tr>
<td>0x480228000 - 0x48022BFFFF</td>
<td>Delay Block SDMMC2</td>
<td></td>
<td>Section 26.4: DLYB registers</td>
</tr>
<tr>
<td>0x480224000 - 0x480227FFF</td>
<td>SDMMC2</td>
<td></td>
<td>Section 60.9: SDMMC registers</td>
</tr>
<tr>
<td>0x480218000 - 0x48021BFFFF</td>
<td>RNG</td>
<td></td>
<td>Section 38.7: RNG registers</td>
</tr>
<tr>
<td>0x480214000 - 0x480217FFFF</td>
<td>HASH</td>
<td></td>
<td>Section 40.7: HASH registers</td>
</tr>
<tr>
<td>0x480210000 - 0x480213FFFF</td>
<td>CRYPTO</td>
<td></td>
<td>Section 39.6: CRYPTO registers</td>
</tr>
<tr>
<td>0x480208000 - 0x48020BFFFF</td>
<td>HSEM</td>
<td></td>
<td>Section 10.4: HSEM registers</td>
</tr>
<tr>
<td>0x480204000 - 0x480207FFFF</td>
<td>PSSI</td>
<td></td>
<td>Section 35.5: PSSI registers</td>
</tr>
<tr>
<td>0x480200000 - 0x480203FFFF</td>
<td>DCMI</td>
<td></td>
<td>Section 34.5: DCMI registers</td>
</tr>
<tr>
<td>0x400400000 - 0x400407FFFF</td>
<td>OTG_HS</td>
<td></td>
<td>Section 62.14: OTG_HS registers</td>
</tr>
<tr>
<td>0x400230000 - 0x400233FFFF</td>
<td>CRC</td>
<td></td>
<td>Section 22.4: CRC registers</td>
</tr>
<tr>
<td>0x400220000 - 0x400223FFFF</td>
<td>ADC1 - ADC2</td>
<td></td>
<td>Section 27.7: ADC common registers</td>
</tr>
<tr>
<td>0x400208000 - 0x40020BFFFF</td>
<td>DMAMUX1</td>
<td></td>
<td>Section 17.6: DMAMUX registers</td>
</tr>
<tr>
<td>0x400204000 - 0x400207FFFF</td>
<td>DMA2</td>
<td></td>
<td>Section 15.5: DMA registers</td>
</tr>
<tr>
<td>0x400200000 - 0x400203FFFF</td>
<td>DMA1</td>
<td></td>
<td>Section 15.5: DMA registers</td>
</tr>
</tbody>
</table>
### Table 6. Register boundary addresses

<table>
<thead>
<tr>
<th>Boundary address</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Register map</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x40017800 - 0x40017FFF</td>
<td>DFSDM1</td>
<td></td>
<td>Section 33.7: DFSDM channel y registers (y=0..7), Section 33.8: DFSDM filter x module registers (x=0..7)</td>
</tr>
<tr>
<td>0x40015C00 - 0x40015FFF</td>
<td>SAI2</td>
<td></td>
<td>Section 56.6: SAI registers</td>
</tr>
<tr>
<td>0x40015800 - 0x40015BFF</td>
<td>SAI1</td>
<td></td>
<td>Section 56.6: SAI registers</td>
</tr>
<tr>
<td>0x40015000 - 0x400153FF</td>
<td>SPI5</td>
<td></td>
<td>Section 55.11: SPI/I2S registers</td>
</tr>
<tr>
<td>0x40014800 - 0x40014BFF</td>
<td>TIM17</td>
<td></td>
<td>Section 45.6: TIM16/TIM17 registers</td>
</tr>
<tr>
<td>0x40014400 - 0x400147FF</td>
<td>TIM16</td>
<td></td>
<td>Section 45.6: TIM16/TIM17 registers</td>
</tr>
<tr>
<td>0x40014000 - 0x400143FF</td>
<td>TIM15</td>
<td></td>
<td>Section 45.5: TIM15 registers</td>
</tr>
<tr>
<td>0x40013400 - 0x400137FF</td>
<td>SPI4</td>
<td></td>
<td>Section 55.11: SPI/I2S registers</td>
</tr>
<tr>
<td>0x40013000 - 0x400133FF</td>
<td>SPI1 / I2S1</td>
<td></td>
<td>Section 55.11: SPI/I2S registers</td>
</tr>
<tr>
<td>0x40011C00 - 0x40011FFF</td>
<td>USART10</td>
<td></td>
<td>Section 53.7: USART registers</td>
</tr>
<tr>
<td>0x40011800 - 0x40011BFF</td>
<td>UART9</td>
<td></td>
<td>Section 53.7: USART registers</td>
</tr>
<tr>
<td>0x40011400 - 0x400117FF</td>
<td>USART6</td>
<td></td>
<td>Section 53.7: USART registers</td>
</tr>
<tr>
<td>0x40011000 - 0x400113FF</td>
<td>USART1</td>
<td></td>
<td>Section 53.7: USART registers</td>
</tr>
<tr>
<td>0x40010400 - 0x400107FF</td>
<td>TIM8</td>
<td></td>
<td>Section 42.4: TIM1/TIM8 registers</td>
</tr>
<tr>
<td>0x40010000 - 0x400103FF</td>
<td>TIM1</td>
<td></td>
<td>Section 42.4: TIM1/TIM8 registers</td>
</tr>
</tbody>
</table>
### Table 6. Register boundary addresses

<table>
<thead>
<tr>
<th>Boundary address</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Register map</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x4000AC00 - 0x4000D3FF</td>
<td>CAN Message RAM</td>
<td>APB1 (CD)</td>
<td>Section 61.5: FDCAN registers</td>
</tr>
<tr>
<td>0x4000A800 - 0x4000ABFF</td>
<td>CAN CCU</td>
<td></td>
<td>Section 61.5: FDCAN registers</td>
</tr>
<tr>
<td>0x4000A400 - 0x4000A7FF</td>
<td>FDCAN</td>
<td></td>
<td>Section 61.5: FDCAN registers</td>
</tr>
<tr>
<td>0x4000A000 - 0x4000A3FF</td>
<td>TT-FDCAN</td>
<td></td>
<td>Section 61.5: FDCAN registers</td>
</tr>
<tr>
<td>0x40009400 - 0x400097FF</td>
<td>MDIOS</td>
<td></td>
<td>Section 59.4: MDIOS registers</td>
</tr>
<tr>
<td>0x40009000 - 0x400093FF</td>
<td>OPAMP</td>
<td></td>
<td>Section 32.6: OPAMP registers</td>
</tr>
<tr>
<td>0x40008800 - 0x40008BFF</td>
<td>SWPMI</td>
<td></td>
<td>Section 58.6: SWPMI registers</td>
</tr>
<tr>
<td>0x40008400 - 0x400087FF</td>
<td>CRS</td>
<td></td>
<td>Section 9.8: CRS registers</td>
</tr>
<tr>
<td>0x40007C00 - 0x40007FFF</td>
<td>UART8</td>
<td></td>
<td>Section 53.7: USART registers</td>
</tr>
<tr>
<td>0x40007800 - 0x40007BFF</td>
<td>UART7</td>
<td></td>
<td>Section 53.7: USART registers</td>
</tr>
<tr>
<td>0x40007400 - 0x400077FF</td>
<td>DAC1</td>
<td></td>
<td>Section 29.7: DAC registers</td>
</tr>
<tr>
<td>0x40006C00 - 0x40006FFF</td>
<td>HDMI-CEC</td>
<td></td>
<td>Section 63.7: HDMI-CEC registers</td>
</tr>
<tr>
<td>0x40005C00 - 0x40005FFF</td>
<td>I2C3</td>
<td></td>
<td>Section 52.7: I2C registers</td>
</tr>
<tr>
<td>0x40005800 - 0x40005BFF</td>
<td>I2C2</td>
<td></td>
<td>Section 52.7: I2C registers</td>
</tr>
<tr>
<td>0x40005400 - 0x400057FF</td>
<td>I2C1</td>
<td></td>
<td>Section 52.7: I2C registers</td>
</tr>
<tr>
<td>0x40005000 - 0x400053FF</td>
<td>UART5</td>
<td></td>
<td>Section 53.7: USART registers</td>
</tr>
<tr>
<td>0x40004C00 - 0x40004FFFF</td>
<td>UART4</td>
<td></td>
<td>Section 53.7: USART registers</td>
</tr>
<tr>
<td>0x40004800 - 0x40004BFF</td>
<td>USART3</td>
<td></td>
<td>Section 53.7: USART registers</td>
</tr>
<tr>
<td>0x40004400 - 0x400047FF</td>
<td>USART2</td>
<td></td>
<td>Section 53.7: USART registers</td>
</tr>
<tr>
<td>0x40004000 - 0x400043FF</td>
<td>SPDIIFRX1</td>
<td></td>
<td>Section 57.5: SPDIIFRX interface registers</td>
</tr>
<tr>
<td>0x40003C00 - 0x40003FFF</td>
<td>SPI3 / I2S3</td>
<td></td>
<td>Section 55.11: SPI/12S registers</td>
</tr>
<tr>
<td>0x40003800 - 0x40003BFF</td>
<td>SPI2 / I2S2</td>
<td></td>
<td>Section 55.11: SPI/12S registers</td>
</tr>
<tr>
<td>0x40002400 - 0x400027FF</td>
<td>LPTIM1</td>
<td></td>
<td>Section 47.7: LPTIM registers</td>
</tr>
<tr>
<td>0x40002000 - 0x400023FF</td>
<td>TIM14</td>
<td></td>
<td>Section 43.4: TIM2/TIM3/TIM4/TIM5 registers</td>
</tr>
<tr>
<td>0x40001C00 - 0x40001FFF</td>
<td>TIM13</td>
<td></td>
<td>Section 43.4: TIM2/TIM3/TIM4/TIM5 registers</td>
</tr>
<tr>
<td>0x40001800 - 0x40001BFF</td>
<td>TIM12</td>
<td></td>
<td>Section 43.4: TIM2/TIM3/TIM4/TIM5 registers</td>
</tr>
<tr>
<td>0x40001400 - 0x400017FF</td>
<td>TIM7</td>
<td></td>
<td>Section 46.4: TIM6/TIM7 registers</td>
</tr>
<tr>
<td>0x40001000 - 0x400013FF</td>
<td>TIM6</td>
<td></td>
<td>Section 46.4: TIM6/TIM7 registers</td>
</tr>
<tr>
<td>0x40000C00 - 0x40000FFF</td>
<td>TIM5</td>
<td></td>
<td>Section 43.4: TIM2/TIM3/TIM4/TIM5 registers</td>
</tr>
<tr>
<td>0x40000800 - 0x40000BFF</td>
<td>TIM4</td>
<td></td>
<td>Section 43.4: TIM2/TIM3/TIM4/TIM5 registers</td>
</tr>
<tr>
<td>0x40000400 - 0x400007FF</td>
<td>TIM3</td>
<td></td>
<td>Section 43.4: TIM2/TIM3/TIM4/TIM5 registers</td>
</tr>
<tr>
<td>0x40000000 - 0x400003FF</td>
<td>TIM2</td>
<td></td>
<td>Section 43.4: TIM2/TIM3/TIM4/TIM5 registers</td>
</tr>
</tbody>
</table>

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

The STM32H7A3/7B3 and STM32H7B0 devices feature:

- Up to 1184 Kbytes of System SRAM
- 128 Kbytes of data TCM RAM
- 64 Kbytes of instruction TCM RAM
- 4 Kbytes of backup SRAM

The embedded system SRAM is divided into up to six blocks:

- **CD domain, AXI SRAM:**
  - AXI SRAM1 mapped at address 0x2400 0000
  - AXI SRAM2 mapped at address 0x2404 0000
  - AXI SRAM3 mapped at address 0x240A 0000
  All AXI SRAMs can be accessed by all system masters through the AXI bus matrix except BDMA1 and BDMA2.

- **CD domain, AHB SRAM:**
  - AHB SRAM1 mapped at address 0x3000 0000
  - AHB SRAM2 mapped at address 0x3001 0000
  All the AHB SRAMs can be accessed by all system masters through the CD AHB matrix except for BDMA2.

- **SRD domain, SRD SRAM:**
  - SRD SRAM is mapped at address 0x3800 0000 and accessible by most of system masters through SRD domain AHB matrix.

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

The AHB masters can read/write-access an SRAM section concurrently with the OTG_HS peripheral accessing another SRAM section.

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

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

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

The backup RAM is mapped at the address 0x3880 0000 and is accessible by most of the system masters through SRD domain AHB matrix. With a battery connected to the V_{BAT} pin, the backup SRAM can be used to retain data during low-power mode (Standby and V_{BAT} mode).

**Error code correction (ECC)**

ITCM-RAM data are protected by ECC:

- 8 ECC bits are added per 64-bit word for ITCM-RAM.
The ECC mechanism is based on the SECDED algorithm. It supports single- and double-error correction.

2.5 Flash memory overview

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

The Flash memory is organized as follows:
- For STM32H7A3/7B3: two main memory block divided into sectors.
- For STM32H7B0: one main memory block divided into sectors.
- An information block:
  - System memory from which the device boots in System memory boot mode
  - Option bytes to configure read and write protection, BOR level, watchdog software/hardware and reset when the device is in Standby or Stop mode.

Refer to Section 4: Embedded Flash memory (FLASH) for more details.

2.6 Boot configuration

In the STM32H7A3/7B3 and STM32H7B0, two different boot areas can be selected through the BOOT pin and the boot base address programmed in the BOOT_ADD0 and BOOT_ADD1 option bytes as shown in the Table 7.

<table>
<thead>
<tr>
<th>Boot mode selection</th>
<th>Boot area</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOT</td>
<td>Boot address option bytes</td>
</tr>
<tr>
<td>0</td>
<td>BOOT_ADD0[15:0]</td>
</tr>
<tr>
<td>1</td>
<td>BOOT_ADD1[15:0]</td>
</tr>
</tbody>
</table>

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

The BOOT pin is also re-sampled when the device exits the Standby mode. Consequently, they must be kept in the required Boot mode configuration when the device is in the Standby mode.

After startup delay, the selection of the boot area is done before releasing the processor reset.
The BOOT_ADD0 and BOOT_ADD1 address option bytes allows to program any boot memory address from 0x0000 0000 to 0x3FFF 0000 which includes:

- All Flash address space
- All RAM address space: ITCM, DTCM RAMs and SRAMs
- The TCM-RAM

The BOOT_ADD0 / BOOT_ADD1 option bytes can be modified after reset in order to boot from any other boot address after next reset.

If the programmed boot memory address is out of the memory mapped area or a reserved area, the default boot fetch address is programmed as follows:

- Boot address 0: FLASH at 0x0800 0000
- Boot address 1: ITCM-RAM at 0x0000 0000

When the Flash level 2 protection is enabled, only boot from Flash memory is available. If the boot address already programmed in the BOOT_ADD0 / BOOT_ADD1 option bytes is out of the memory range or belongs to the RAM address range, the default fetch will be forced from Flash memory at address 0x0800 0000.

**Embedded bootloader**

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

- USART1 on PA9/PA10 pins, USART2 on PA3/2pins, and USART3 on PB10/PB11/PD8/PD9 pins
- I2C1 on PB6/9 pins, I2C2 on PF0/PF1 pins, I2C3 on PA8/PC9 pins, and I2C4 on PD12/13 pins
- USB OTG FS in Device mode (DFU) on PA11/PA12 pins
- SPI1 on PA7/6/5/4 pins, SPI2 on PI3/2/1/0 pins, SPI3 on PC12/11/10/PA15pins, and SPI4 on PE14/13/12/11 pins
- FDCAN1 on PD0/PD1/PH13/PH14

For additional information, refer to the application note AN2606.
3 RAM ECC monitoring (RAMECC)

3.1 Introduction

The STM32H7A3/7B3 and STM32H7B0 devices feature a RAM ECC monitoring unit (RAMECC). It provides a mean for application software to verify ECC status and execute service routines when an error occurs.

3.2 RAMECC main features

TCM and Cache SRAMs are protected by ECC. The ECC mechanism is based on the SECDED algorithm. It supports single- and double-error detection, as well as single-error correction:

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

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

The RAMECC monitoring unit includes the following features:

- RAM ECC monitoring per domain
- RAM failing address/data identification

3.3 RAMECC functional description

3.3.1 RAMECC block diagram

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

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

Note: All the RAM ECC controllers are always enabled.

Figure 3 describes the implementation of RAM ECC controllers.
A RAM ECC monitoring area is defined for the CD domain (see Section 3.3.3: RAMECC monitor mapping. The RAMECC allows the collection of ECC diagnostic events and provides a mean for the CPU to verify the ECC status.

Figure 4 shows the connection schematic between the RAM ECC controller and the RAMECC monitoring unit.
3.3.2 RAMECC internal signals

*Table 8* gives the list of the internal signals that control the RAMECC unit.

![Table 8. RAMECC internal input/output signals](image)

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

3.3.3 RAMECC monitor mapping

STM32H7A3/7B3 and STM32H7B0 devices features one RAMECC monitoring unit. The inputs from the ECC controllers are mapped as described in *Table 9*. The RAM ECC event monitoring status and configuration registers are described in *Section 3.4: RAMECC registers*.

![Table 9. ECC controller mapping](image)

<table>
<thead>
<tr>
<th>RAMECC unit</th>
<th>Monitor number</th>
<th>SRAM ECC event monitoring status and configuration registers</th>
<th>Size in Kbytes</th>
<th>Address Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD domain RAMECC unit</td>
<td>1</td>
<td>ITCM-RAM ECC monitoring unit</td>
<td>64</td>
<td>0x20</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>DTCM-RAM ECC monitoring unit</td>
<td>64</td>
<td>0x40</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>D1TCM</td>
<td>64</td>
<td>0x60</td>
</tr>
</tbody>
</table>
3.4 RAMECC registers

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

3.4.1 RAMECC interrupt enable register (RAMECC_IER)

Address offset: 0x00
Reset value: 0x0000 0000

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

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

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

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

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

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

Address offset: 0x20 * x  
Reset value: 0x0000 0000  
x is the ECC monitoring unit number

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

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

Bits 31:6  Reserved, must be kept at reset value.

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

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

- **Bit 3 ECCDEIE**: ECC double error interrupt enable  
  When ECCDEIE bit is set to 1, monitor x generates an interrupt when an ECC double detection error occurs during a read operation from RAM.  
  0: interrupt generated if an ECC double detection error occurs  
  1: interrupt generated if an ECC double detection error occurs

- **Bit 2 ECCSEIE**: ECC single error interrupt enable  
  When ECCSEIE bit is set to 1, monitor x generates an interrupt when an ECC single error occurs during a read operation from RAM.  
  0: interrupt generated when an ECC single error occurs  
  1: interrupt generated when an ECC single error occurs

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

### 3.4.3 RAMECC monitor x status register (RAMECC_MxSR)

Address offset: 0x24 + 0x20 * (x - 1), (x= ECC monitoring unit number)  
Reset value: 0x0000 0000
3.4.4 RAMECC monitor x failing address register (RAMECC_MxFAR)

Address offset: 0x28 + 0x20 * (x-1), (x= ECC monitoring unit number)
Reset value: 0x0000 0000

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

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

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

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

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

3.4.5 RAMECC monitor x failing data low register (RAMECC_MxFDRL)

Address offset: 0x2C + 0x20 * (x-1), (x= ECC monitoring unit number)
Reset value: 0x00000 0000

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

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

---

142/2974 RM0455 Rev 3
3.4.6 RAMECC monitor x failing data high register (RAMECC_MxFDRH)

Address offset: 0x30 + 0x20 * (x-1), (x= ECC monitoring unit number)
Reset value: 0x0000 0000

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

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

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

Address offset: 0x34 + 0x20 * (x-1), (x= ECC monitoring unit number)
Reset value: 0x0000 0000

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

Bits 31:0 FDATAH[31:0]: Failing data high (64-bit memory)
When an ECC error occurs the FDATAH bitfield contains the MSB part of the data that generated the error.

Note: This register is reserved in case of 32-bit word SRAM.

Bits 31:0 FEC[31:0]: Failing error code
When an ECC error occurs the FEC bitfield contains the ECC failing code that generated the error.
### 3.4.8 RAMECC register map

Table 10. RAMECC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name reset value</th>
<th>Register size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>RAMECC_IER</td>
<td>31</td>
</tr>
<tr>
<td>0x20 * x (x = monitoring unit number)</td>
<td>RAMECC_MxCR</td>
<td>0x00</td>
</tr>
<tr>
<td>0x24+0x20 *(x -1) (x = monitoring unit number)</td>
<td>RAMECC_MxSR</td>
<td>0x20</td>
</tr>
<tr>
<td>0x28+0x20 *(x -1) (x = monitoring unit number)</td>
<td>RAMECC_MxFAR</td>
<td>FADD[31:0]</td>
</tr>
<tr>
<td>0x2C+0x20 *(x -1) (x = monitoring unit number)</td>
<td>RAMECC_MxFDRL</td>
<td>FDATAL[31:0]</td>
</tr>
<tr>
<td>0x30+0x20 *(x -1) (x = monitoring unit number)</td>
<td>RAMECC_MxFDRH</td>
<td>FDATAH[31:0]</td>
</tr>
<tr>
<td>0x34+0x20 *(x -1) (x = monitoring unit number)</td>
<td>RAMECC_MxFECR</td>
<td>FEC[31:9]</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
4 Embedded Flash memory (FLASH)

4.1 Introduction

The embedded Flash memory (FLASH) manages the accesses of any master to the embedded non-volatile memory, that is up to 2 Mbytes. It implements the read, program and erase operations, error corrections as well as various integrity and confidentiality protection mechanisms.

The embedded Flash memory manages the automatic loading of non-volatile user option bytes at power-on reset, and implements the dynamic update of these options.

The embedded Flash memory also features a one-time-programmable (OTP) area and a read-only area provisioned by STMicroelectronics during the product manufacturing.

4.2 FLASH main features

- Up to 2 Mbytes of non-volatile memory divided into two banks of up to 1 Mbyte each
- Flash memory read operations supporting multiple length (64 bits, 32bits, 16bits or one byte)
- Flash memory programming by 128 bits (user area) and 16 bits (OTP area)
- 8-Kbytes sector erase, bank erase and dual-bank mass erase
- Dual-bank organization supporting (only available on SMT32H7A3/7B3 devices):
  - simultaneous operations: two read/program/erase operations executed in parallel on both banks
  - Bank swapping: the address mapping of the user Flash memory of each bank can be swapped, along with the corresponding registers.
- Error Code Correction (ECC): one error detection/correction or two error detections per 128-bit Flash word using 9 ECC bits
- Cyclic redundancy check (CRC) hardware module
- User configurable non-volatile option bytes
- Flash memory enhanced protections, activated by option bytes
  - Read protection (RDP), preventing unauthorized Flash memory dump to safeguard sensitive application code
  - Sector group write-protection (WRPSG), protecting up to 32 group of 4 sectors (32 Kbytes) per bank
  - Two proprietary code readout protection (PCROP) areas (one per user Flash bank). When enabled, this area is execute-only.
  - Two secure-only areas (one per user Flash bank). When enabled this area is accessible only if the STM32 microcontroller operates in Secure access mode (only available on STM32H7B0 and STM32H7B3 devices).
- 1-Kbyte one-time programmable (OTP) area
- Read-only area provisioned by STMicroelectronics
- Read command queue to streamline Flash operations
4.3 FLASH functional description

4.3.1 FLASH block diagram

*Figure 5* shows the embedded Flash memory block diagram.

*Note:* Bank 2 is only available on STM32H7A3/7B3 devices.
4.3.2 FLASH internal signals

*Table 11* describes a list of the useful to know internal signals available at embedded Flash memory level. These signals are not available on the microcontroller pads.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_ck</td>
<td>Input</td>
<td>CPU domain bus clock (embedded Flash memory AXI interface clock)</td>
</tr>
<tr>
<td>po_rst</td>
<td>Input</td>
<td>Power on reset</td>
</tr>
<tr>
<td>sys_rst</td>
<td>Input</td>
<td>CPU domain system reset</td>
</tr>
<tr>
<td>flash_it</td>
<td>Output</td>
<td>Embedded Flash memory interrupt request</td>
</tr>
</tbody>
</table>

4.3.3 FLASH architecture and integration in the system

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

The embedded Flash memory offers three different interconnect interfaces:

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

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

The embedded Flash memory microarchitecture is shown in *Figure 6*. 
Behind the system interfaces, the embedded Flash memory implements various command queues and buffers to perform Flash read, write and erase operations with maximum efficiency.

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

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

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

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

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

Note: Bank 2 is only available on STM32H7A3/7B3 devices.
4.3.4 Flash memory architecture and usage

Flash memory architecture

Figure 7 shows the non-volatile memory organization supported by the embedded Flash memory.

Figure 7. Embedded Flash memory organization

Note: On STM32H7B0 devices, bank 2 is not available and bank 1 contains 16 sectors of 8 Kbytes each.

The embedded Flash non-volatile memory is composed of:

- For STM32H7B3xx and STM32H7A3xl devices: a 2-Mbyte main memory block, organized as two banks of 1 Mbyte each.
- For STM32H7A3xG devices: a 1-Mbyte main memory block, organized as two banks of 512 Kbyte each.
- For STM32H7B0 devices: a 128-Kbyte main memory block, organized in a single bank.
- Each bank is in turn divided in up to 128 sectors of 8 Kbytes each, and features Flash-word rows of 128 bits + 9 bits of ECC per word.
- A system memory block of 128 Kbytes, divided into two 64-Kbyte banks. Each bank is in turn divided eight 8-Kbyte sectors. The system Flash memory is ECC protected.
- A set of non-volatile option bytes loaded at reset by the embedded Flash memory and accessible by the application software only through the AHB configuration register interface.
- A 1 Kbyte one-time-programmable (OTP) area that can be written only once by the application software.
- A 512-byte read-only area that can be written only by STMicroelectronics.
Note: The system Flash memory on bank 1 and bank 2 cannot be programmed nor erased by any application.
Since the OTP area has no specific protection, it must not be used to store confidential information.

The overall Flash memory architecture and its corresponding access interface is summarized in Table 12, Table 13 and Table 14.

Table 12. Flash memory organization (STM32H7A3xI/7B3xI devices)

<table>
<thead>
<tr>
<th>Flash memory area</th>
<th>Address range</th>
<th>Size (bytes)</th>
<th>Region name</th>
<th>Access interface</th>
<th>SSN1/2(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User main memory</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank 1</td>
<td>0x0800 0000-0x0800 1FFF</td>
<td>8 K</td>
<td>Sector 0</td>
<td>AXI port</td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td>0x0800 2000-0x0800 3FFF</td>
<td>8 K</td>
<td>Sector 1</td>
<td></td>
<td>0x01</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>0x080F E000-0x080F FFFF</td>
<td>8 K</td>
<td>Sector 127</td>
<td></td>
<td>0x7F</td>
</tr>
<tr>
<td>Bank 2</td>
<td>0x0810 0000-0x0810 1FFF</td>
<td>8 K</td>
<td>Sector 0</td>
<td>AXI port</td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td>0x0810 2000-0x0810 3FFF</td>
<td>8 K</td>
<td>Sector 1</td>
<td></td>
<td>0x01</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>0x081F E000-0x081F FFFF</td>
<td>8 K</td>
<td>Sector 127</td>
<td></td>
<td>0x7F</td>
</tr>
<tr>
<td><strong>System memory</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank 1</td>
<td>0x1FF0 0000-0x1FF0 1FFF</td>
<td>8 K</td>
<td>Sys Sector 0 (read-only)</td>
<td>AXI port</td>
<td>N/A(2)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>0x1FF0 E000-0x1FF0 FFFF</td>
<td>8 K</td>
<td>Sys Sector 7 (read-only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank 2</td>
<td>0x1FF1 0000-0x1FF1 1FFF</td>
<td>8 K</td>
<td>Sys Sector 0 (read-only)</td>
<td>AXI port</td>
<td>N/A(2)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>0x1FF1 E000-0x1FF1 FFFF</td>
<td>8 K</td>
<td>Sys Sector 7 (read-only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Special region</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank 1</td>
<td>0x08FF F000-0x08FF F3FF</td>
<td>1 K</td>
<td>OTP area</td>
<td>AHB system port</td>
<td>N/A(2)</td>
</tr>
<tr>
<td></td>
<td>0x08FF F800-0x08FF F9FF</td>
<td>512</td>
<td>Read-only area</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Option bytes</strong></td>
<td>N/A</td>
<td>-</td>
<td>User option bytes</td>
<td>Registers only</td>
<td>N/A(2)</td>
</tr>
</tbody>
</table>

1. SSN1/2 contain the target sector number for an erase operation. See Section 4.3.10 for details.
2. Cannot be erased by application software.
### Table 13. Flash memory organization (STM32H7B0 devices)

<table>
<thead>
<tr>
<th>Flash memory area</th>
<th>Address range</th>
<th>Size (bytes)</th>
<th>Region name</th>
<th>Access interface</th>
<th>SSN1(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User main memory</td>
<td>Bank 1</td>
<td>0x0800 0x0000-0x0800 1FFF</td>
<td>8 K</td>
<td>Sector 0</td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0800 2000-0x0800 3FFF</td>
<td>8 K</td>
<td>Sector 1</td>
<td>0x01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AXI port</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0801 E000-0x0801 FFFF</td>
<td>8 K</td>
<td>Sector 15</td>
<td>0x0F</td>
</tr>
<tr>
<td>System memory</td>
<td>Bank 1</td>
<td>0x1FF0 0x0000-0x1FF0 FFFF</td>
<td>64 K</td>
<td>System Flash (read-only)</td>
<td>N/A(2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AXI port</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bank 2</td>
<td>0x1FF1 0x0000-0x1FF1 FFFF</td>
<td>64 K</td>
<td>System Flash (read-only)</td>
<td>N/A(2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Option bytes</td>
<td>N/A</td>
<td>User option bytes</td>
<td>Registers only(3)</td>
<td>N/A(2)</td>
</tr>
</tbody>
</table>

1. SSN1 contains the target sector number for an erase operation. See Section 4.3.10 for details.
2. Cannot be erased by application software.
3. On STM32H750 devices, only bank 1 option byte registers are applicable. Bank 2 option byte registers must be kept at their reset value.

### Table 14. Flash memory organization (STM32H7A3xG devices)

<table>
<thead>
<tr>
<th>Flash memory area</th>
<th>Address range</th>
<th>Size (bytes)</th>
<th>Region name</th>
<th>Access interface</th>
<th>SSN1/2(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User main memory</td>
<td>Bank 1</td>
<td>0x0800 0x0000-0x0800 1FFF</td>
<td>8 K</td>
<td>Sector 0</td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0800 2000-0x0800 3FFF</td>
<td>8 K</td>
<td>Sector 1</td>
<td>0x01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AXI port</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0807 E000-0x0807 FFFF</td>
<td>8 K</td>
<td>Sector 63</td>
<td>0x3F</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bank 2</td>
<td>0x0810 0x0000-0x0810 1FFF</td>
<td>8 K</td>
<td>Sector 0</td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0810 2000-0x0810 3FFF</td>
<td>8 K</td>
<td>Sector 1</td>
<td>0x01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AXI port</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0817 E000-0x0817 FFFF</td>
<td>8 K</td>
<td>Sector 63</td>
<td>0x3F</td>
</tr>
</tbody>
</table>
Table 14. Flash memory organization (STM32H7A3xG devices) (continued)

<table>
<thead>
<tr>
<th>Flash memory area</th>
<th>Address range</th>
<th>Size (bytes)</th>
<th>Region name</th>
<th>Access interface</th>
<th>SSN1/2(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>System memory</td>
<td>Bank 1</td>
<td>0x1FF0 0000-0x1FF0 FFFF</td>
<td>64 K System Flash (read-only)</td>
<td>AXI port</td>
<td>N/A(2)</td>
</tr>
<tr>
<td></td>
<td>Bank 2</td>
<td>0x1FF1 0000-0x1FF1 FFFF</td>
<td>64 K System Flash (read-only)</td>
<td></td>
<td>N/A(2)</td>
</tr>
<tr>
<td>Option bytes</td>
<td>N/A</td>
<td>-</td>
<td>User option bytes</td>
<td>Registers only</td>
<td>N/A(2)</td>
</tr>
</tbody>
</table>

1. SSN1/2 contain the target sector number for an erase operation. See Section 4.3.10 for details.
2. Cannot be erased by application software.
Partition usage

*Figure 8* shows how the embedded Flash memory is used both by STMicroelectronics and the application software.

**Figure 8. Embedded Flash memory usage**

1. Bank 2 is available only on STM32H7A3/7B3 devices.
2. Bank swapping is supported only on STM32H7A3/7B3 devices.

User and system memories are used differently according to whether the microcontroller is configured by the application software in Standard mode or in Secure access mode (STM32H7B0 and STM32H7B3 devices only). This selection is done through the SECURITY option bit (see *Section 4.4.6*):

- In Standard mode, the user memory contains the application code and data, while the system memory is loaded with the STM32 bootloader. When a reset occurs, the core
jumps to the boot address configured through the BOOT pin and the BOOT_ADD0/1 option bytes.

- In Secure access mode, dedicated libraries can be used for secure boot. They are located in user Flash and system Flash memory:
  - ST libraries in system Flash memory assist the application software boot with special features such as secure boot and secure firmware install (SFI).
  - Application secure libraries in user Flash memory are used for secure firmware update (SFU).

In Secure access mode, the microcontroller always boots into the secure bootloader code (unique entry point). Then, if no secure services are required, this code securely jumps to the requested boot address configured through the BOOT pin and the option bytes, as shown in Figure 8 (see Section 5: Secure memory management (SMM) for details).

**Note:** For more information on option byte setup for boot, refer to Section 4.4.7.

Additional partition usage is the following:

- The option bytes are used by STMicroelectronics and by the application software as non-volatile product options (e.g. boot address, protection configuration and reset behaviors).
- The OTP area is used by the application software. This area is virgin when the device is delivered by STMicroelectronics.
- The read-only area is programmed by STMicroelectronics during manufacturing to contain unique device ID and product information.

**Note:** For further information on STM32 bootloader flashing by STMicroelectronics, refer to application note AN2606 “STM32 microcontroller system memory boot mode” available from www.st.com.

**Bank swapping (STM32H7A3/7B3 only)**

As shown in Figure 8, the embedded Flash memory offers a bank swapping feature that can be configured through the SWAP_BANK bit, always available for the application. For more information please refer to Section 4.3.15.

**4.3.5 FLASH system performance enhancements**

The embedded Flash memory does not implement a local cache memory to read or write data. Read command queues (one per bank) are used instead to enhance Flash operations.

**4.3.6 FLASH data protection schemes**

*Figure 9* gives an overview of the protection mechanisms supported by the embedded Flash memory. A PCROP and a secure-only area can be defined for each bank. The properties of these protected areas are detailed in Section 4.5. PCROP is supported by all devices, while the secure-only areas are available only on STM32H7B0/B3.
Figure 9. FLASH protection mechanisms

Note: For STM32H7A3XG devices, bank 1 and bank 2 are limited to 64 sectors.
For STM32H7B0 devices, bank 1 is limited to 16 sectors.
Bank 2 is available only on STM32H7A3/7B3 devices.

4.3.7 Overview of FLASH operations

Read operations

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

The embedded Flash memory supports read-while-write operations provided the read and write operations target different banks. Similarly read-while-read operations are supported when two read operations target different banks. Read-while-write and read-while-read are supported only on STM32H7A3/7B3 devices.

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

For more details on read operations, refer to Section 4.3.8: FLASH read operations.

Program/erase operations

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

- Single Flash word write (128-bit granularity for user Flash memory, 16-bit granularity for OTP area), with the possibility for the application to force-write a user Flash word with less than 128 bits in user Flash memory
- Single sector erase
- Bank erase (single or dual)
- Option byte update
Thanks to its dual bank architecture, the embedded Flash memory can perform any of the above write or erase operation on one bank while a read or another program/erase operation is executed on the other bank.

Dual bank and related operations are supported only on STM32H7A3/7B3 devices.

Note: Program and erase operations are subject to the various protection that could be set on the embedded Flash memory, such as write protection and global readout protection (see next sections for details).

For more details refer to Section 4.3.9: FLASH program operations and Section 4.3.10: FLASH erase operations.

Protection mechanisms

The embedded Flash memory supports different protection mechanisms:

- Global readout protection (RDP)
- Proprietary code readout protection (PCROP)
- Write protection
- Secure access only protection

For more details refer to Section 4.5: FLASH protection mechanisms.

Option byte loading

Under specific conditions, the embedded Flash memory reliably loads the non-volatile option bytes stored in non-volatile memory, thus enforcing boot and security options to the whole system when the embedded Flash memory becomes functional again. For more details refer to Section 4.4: FLASH option bytes.

Bank/register swapping (STM32H7A3/7B3 only)

The embedded Flash memory allows swapping bank 1 and bank 2 memory mapping. This feature can be used after a firmware upgrade to restart the microcontroller on the new firmware after a system reset. For more details on the feature, refer to Section 4.3.15: Flash bank and register swapping (STM32H7A3/7B3 only).

4.3.8 FLASH read operations

Read operation overview

The embedded Flash memory supports, for each memory bank, the execution of one read command while two are waiting in the read command queue. Multiple read access types are also supported as defined in Section 4.3.3: FLASH architecture and integration in the system.

The read commands to each bank are associated with a 128-bit read data buffer. These commands can be issued either by the AHB (OTP or read-only areas) or by the AXI interface (user Flash or system Flash memory).
The embedded Flash memory can perform single error correction and double error detection while read operations are being executed (see Section 4.3.12: Flash memory error protections).

The AXI interface read channel operates as follows:

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

The AHB system interface operates as follows:

- If the application reads an OTP data that has not been previously written, a double ECC error is reported and only 1’s are returned (see Section 4.3.13: FLASH one-time programmable area for details)
- Until the read request has been served, the embedded Flash memory stalls the AHB bus and consequently the master that issued that request.

The Read pipeline architecture is summarized in Figure 10.

For more information on bus interfaces, refer to Section 4.3.3: FLASH architecture and integration in the system.

![Figure 10. FLASH read pipeline architecture](image)

Note: Flash bank 2 is available only on STM32H7A3/7B3 devices.
Single read sequence

The recommended simple read sequence is the following:

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

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

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

Adjusting read timing constraints

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

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

Table 15 shows the correspondence between the number of wait states (LATENCY), the programming delay parameter (WRHIGHFREQ), the embedded Flash memory clock frequency and its supply voltage ranges.

<table>
<thead>
<tr>
<th>Number of wait states (LATENCY)</th>
<th>Programming delay (WRHIGH FREQ)</th>
<th>AXI Interface clock frequency v.s. V_{CORE} range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 WS (1 FLASH clock cycle)</td>
<td>00</td>
<td>VOS3 range 0.95 V - 1.05 V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VOS2 range 1.05 V - 1.15 V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VOS1 range 1.15 V - 1.25 V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VOS0 range 1.25 V - 1.35 V</td>
</tr>
<tr>
<td>1 WS (2 FLASH clock cycles)</td>
<td>00</td>
<td>22 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>34 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>38 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>42 MHz</td>
</tr>
<tr>
<td>2 WS (3 FLASH clock cycles)</td>
<td>01</td>
<td>44 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>68 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>76 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>84 MHz</td>
</tr>
<tr>
<td>3 WS (4 FLASH clock cycles)</td>
<td></td>
<td>66 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>102 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>114 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>126 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>88 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>136 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>152 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>168 MHz</td>
</tr>
</tbody>
</table>
Adjusting system frequency

After power-on, the embedded Flash memory is clocked by the 64 MHz high-speed internal oscillator (HSI), with a voltage range set at a voltage scaling value of VOS3. As a result, a conservative 3 wait-state latency is specified in FLASH_ACR register (see Table 15).

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

To increase the embedded Flash memory clock source frequency:
1. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in the FLASH_ACR register, as described in Table 15.
2. Check that the new number of wait states is taken into account by reading back the FLASH_ACR register.
3. Modify the embedded Flash memory clock source and/or the AXI bus clock prescaler in the RCC_CFGR register of the reset and clock controller (RCC).
4. Check that the new embedded Flash memory clock source and/or the new AXI bus clock prescaler value are taken into account by reading back the embedded Flash memory clock source status and/or the AXI bus prescaler value in the RCC_CFGR register of the reset and clock controller (RCC).

To decrease the embedded Flash memory clock source frequency:
1. Modify the embedded Flash memory clock source and/or the AXI bus clock prescaler value in the RCC_CFGR register of reset and clock controller (RCC).
2. Check that the embedded Flash memory new clock source and/or the new AXI bus clock prescaler value are taken into account by reading back the embedded Flash

---

Table 15. FLASH recommended number of wait states and programming delay (continued)

<table>
<thead>
<tr>
<th>Number of wait states (LATENCY)</th>
<th>Programming delay (WRHIGH FREQ)</th>
<th>AXI Interface clock frequency v.s. $V_{CORE}$ range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>VOS3 range 0.95 V - 1.05 V</td>
</tr>
<tr>
<td>4 WS (5 FLASH clock cycles)</td>
<td>10</td>
<td>88 MHz</td>
</tr>
<tr>
<td>5 WS (6 FLASH clock cycles)</td>
<td></td>
<td>88 MHz</td>
</tr>
<tr>
<td>6 WS (7 FLASH clock cycles)</td>
<td>11</td>
<td>88 MHz</td>
</tr>
<tr>
<td>7 WS (8 FLASH clock cycles)</td>
<td>11</td>
<td>88 MHz</td>
</tr>
</tbody>
</table>
memory clock source status and/or the AXI interface prescaler value in the RCC_CFGR register of reset and clock controller (RCC).

3. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in FLASH_ACR register, as described in Table 15.

4. Check that the new number of wait states has been taken into account by reading back the FLASH_ACR register.

**Error code correction (ECC)**

The embedded Flash memory embeds an error correction mechanism. Single error correction and double error detection are performed for each read operation. For more details, refer to Section 4.3.12: Flash memory error protections.

**Read errors**

When the ECC mechanism is not able to correct the read operation, the embedded Flash memory reports read errors as described in Section 4.7.6: Error correction code error (SNECCERR/DBECCERR).

**Read interrupts**

See Section 4.8: FLASH interrupts for details.

### 4.3.9 FLASH program operations

**Program operation overview**

The virgin state of each non-volatile memory bitcell is 1. The embedded Flash memory supports programming operations that can change (reset) any memory bitcell to 0. However these operations do not support the return of a bit to its virgin state. In this case an erase operation of the entire sector is required.

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

*Note:* The application can decide to write as little as 8 bits to a 128 Flash word. In this case, a force-write mechanism to the 128 bits + ECC is used (see FW1/2 bit of FLASH_CR1/2 register).

System Flash memory bank 1 and bank 2 cannot be written by the application software.

It is not recommended to overwrite a Flash word that is not virgin. The result may lead to an inconsistent ECC code that will be systematically reported by the embedded Flash memory, as described in Section 4.7.6: Error correction code error (SNECCERR/DBECCERR).

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

*Note:* The OTP area is typically write-protected on the final product, as described in Section 4.3.13: FLASH one-time programmable area.

Erase operations to the OTP area are not supported.
The AXI interface write channel operates as follows:
- A 128-bit write data buffer is associated with the AXI interface. It supports multiple write access types (64 bits, 32 bits, 16 bits and 8 bits).
- When the write queue is full, any new AXI write request stalls the bus write channel interface and consequently the master that issued that request.

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

The write pipeline architecture is described in Figure 11.

For more information on bus interfaces, refer to Section 4.3.3: FLASH architecture and integration in the system.

Figure 11. FLASH write pipeline architecture

Note: Flash bank 2 is available only on STM32H7A3/7B3 devices.

Managing write protections
Before programming a user sector, the application software must check the protection of the targeted Flash memory area.
The embedded Flash memory checks the protection properties of the write transaction target at the output of the write queue buffer, just before the effective write operation to the non-volatile memory:

- If a write protection violation is detected, the write operation is canceled and write protection error (WRPERR1/2) is raised in FLASH_SR1/2 register.
- If the write operation is valid, the 9-bit ECC code is concatenated to the 128 bits of data and the write to non-volatile memory is effectively executed.

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

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

- If a write protection violation is detected, the write operation is canceled and write protection error (WRPERR1/2) is raised in FLASH_SR1/2 register.
- If the write operation is valid, the 6-bit ECC code is concatenated to the 16 bits of data and the write to non-volatile memory is effectively executed.

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

**Monitoring ongoing write operations**

The application software can use three status flags located in FLASH_SR1/2 in order to monitor ongoing write operations. Those status are available for each bank.

- **BSY1/2**: this bit indicates that an effective write, erase or option byte change operation is ongoing to the non-volatile memory.
- **QW1/2**: this bit indicates that a write, erase or option byte change operation is pending in the write queue or command queue buffer. It remains high until the write operation is complete. It supersedes the BSY1/2 status bit.
- **WBNE1/2**: this bit indicates that the embedded Flash memory is waiting for new data to complete the 128-bit write buffer. In this state the write buffer is not empty. It is reset as soon as the application software fills the write buffer, force-writes the operation using FW1/2 bit in FLASH_CR1/2, or disables all write operations in the corresponding bank.

**Enabling write operations**

Before programming the user Flash memory in bank 1 (respectively bank 2), the application software must make sure that PG1 bit (respectively PG2) is set to 1 in FLASH_CR1 (respectively FLASH_CR2). If it is not the case, an unlock sequence must be used (see Section 4.5.1: FLASH configuration protection) and the PG1/2 bit must be set.

When the option bytes need to be modified or a mass erase needs to be started, the application software must make sure that FLASH_OPTCR is unlocked. If it is not the case, an unlock sequence must be used (see Section 4.5.1: FLASH configuration protection).

**Note:** The application software must not unlock a register that is already unlocked, otherwise this register will remain locked until next system reset.
If needed, the application software can update the programming delay as described at the end of this section.

**Single write sequence**

The recommended single write sequence in bank 1/2 is the following:

1. Unlock the FLASH_CR1/2 register, as described in Section 4.5.1: FLASH configuration protection (only if register is not already unlocked).
2. Enable write operations by setting the PG1/2 bit in the FLASH_CR1/2 register.
3. Check the protection of the targeted memory area.
4. Write one Flash-word corresponding to 16-byte data starting at 16-byte aligned address.
5. Check that QW1 (respectively QW2) has been raised and wait until it is reset to 0.

If step 4 is executed incrementally (e.g. byte per byte), the write buffer can become partially filled. In this case the application software can decide to force-write what is stored in the write buffer by using FW1/2 bit in FLASH_CR1/2 register. In this particular case, the unwritten bits are automatically set to 1. If no bit in the write buffer is cleared to 0, the FW1/2 bit has no effect.

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

Any write access requested while the PG1/2 bit is cleared to 0 is rejected. In this case, no error is generated on the bus, but the PGSERR1/2 flag is raised.

Clearing the programming sequence error (PGSERR) and inconsistency error (INCERR) is mandatory before attempting a write operation (see Section 4.7: FLASH error management for details).

**Adjusting programming timing constraints**

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

The user must therefore trim the optimal programming delay through the WRHIGHFREQ parameter in the FLASH_ACR register. Refer to Table 15 in Section 4.3.8: FLASH read operations for the recommended programming delay depending on the embedded Flash memory clock frequency.

FLASH_ACR configuration register is common to both banks.

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

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

**Programming errors**

When a program operation fails, an error can be reported as described in Section 4.7: FLASH error management.

**Programming interrupts**

See Section 4.8: FLASH interrupts for details.
4.3.10  FLASH erase operations

Erase operation overview

The embedded Flash memory can perform erase operations on 8-Kbyte user sectors, on one user Flash memory bank or on two user Flash memory banks (i.e. mass erase). The system Flash memory (bank 2 only) can also be erased, but by STMicroelectronics secure firmware only.

Note: System Flash memory bank 1 and read-only/OTP area cannot be erased by the application software.

The erase operation forces all non-volatile bit cells to high state, which corresponds to the virgin state. It clears existing data and corresponding ECC, allowing a new write operation to be performed. If the application software reads back a word that has been erased, all the bits will be read at 1. No ECC error is generated if the word is located in user Flash memory, while a double ECC error is raised if the word is located in read-only or OTP area.

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

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

Erase and security

A user sector can be erased only if it does not contain PCROP, secure-only or write-protected data (see Section 4.5: FLASH protection mechanisms for details). In other words, if the application software attempts to erase a user sector with at least one Flash word that is protected, the sector erase operation is aborted and the WRPERR1/2 flag is raised in the FLASH_SR1/2 register, as described in Section 4.7.2: Write protection error (WRPERR).

The embedded Flash memory allows the application software to perform an erase followed by an automatic protection removal (PCROP, secure-only area and write protection), as described hereafter.

Enabling erase operations

Before erasing a sector in bank 1 (respectively bank 2), the application software must make sure that FLASH_CR1 (respectively FLASH_CR2) is unlocked. If it is not the case, an unlock sequence must be used (see Section 4.5.1: FLASH configuration protection).

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

Similar constraints apply to bank erase requests.
**Flash sector erase sequence**

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

1. Check and clear (optional) all the error flags due to previous programming/erase operation. Refer to Section 4.7: FLASH error management for details.

2. Unlock the FLASH_CR1/2 register, as described in Section 4.5.1: FLASH configuration protection (only if register is not already unlocked).

3. Set the SER1/2 bit and SNB1/2 bitfield in the corresponding FLASH_CR1/2 register. SER1/2 indicates a sector erase operation, while SNB1/2 contains the target sector number.

4. Set the START1/2 bit in the FLASH_CR1/2 register.

5. Wait for the QW1/2 bit to be cleared in the corresponding FLASH_SR1/2 register.

*Note:* If a bank erase is requested simultaneously to the sector erase (BER1/2 bit set), the bank erase operation supersedes the sector erase operation.

**Standard Flash bank erase sequence**

To erase all bank sectors except for those containing secure-only and protected data, proceed as follows:

1. Check and clear (optional) all the error flags due to previous programming/erase operation. Refer to Section 4.7: FLASH error management for details.

2. Unlock the FLASH_CR1/2 register, as described in Section 4.5.1: FLASH configuration protection (only if register is not already unlocked).

3. Set the BER1/2 bit in the FLASH_CR1/2 register corresponding to the targeted bank.

4. Set the START1/2 bit in the FLASH_CR1/2 register to start the bank erase operation. Then wait until the QW1/2 bit is cleared in the corresponding FLASH_SR1/2 register.

*Note:* BER1/2 and START1/2 bits can be set together, so above steps 3 and 4 can be merged. If a sector erase is requested simultaneously to the bank erase (SER1/2 bit set), the bank erase operation supersedes the sector erase operation.
Flash bank erase with automatic protection-removal sequence

To erase all bank sectors including those containing secure-only and protected data without performing an RDP regression (see Section 4.5.3), proceed as follows:

1. Check and clear (optional) all the error flags due to previous programming/erase operation. Refer to Section 4.7: FLASH error management for details.
2. Unlock FLASH_OPTCR register, as described in Section 4.5.1: FLASH configuration protection (only if register is not already unlocked).
3. If a PCROP-protected area exists set DMEP1/2 bit in FLASH_PRAR_PRG1/2 register. In addition, program the PCROP area end and start addresses so that the difference is negative, i.e. PROT_AREA_END1/2 < PROT_AREA_START1/2.
4. If a secure-only area exists set DMES1/2 bit in FLASH_SCAR_PRG1/2 register. In addition, program the secure-only area end and start addresses so that the difference is negative, i.e. SEC_AREA_END1/2 < SEC_AREA_START1/2.
5. Set all WRPSGn1/2 bits in FLASH_WPSGN_PRG1/2R to 1 to disable all sector write protection.
6. Unlock FLASH_CR1/2 register, only if register is not already unlocked.
7. Set the BER1/2 bit in the FLASH_CR1/2 register corresponding to the target bank.
8. Set the START1/2 bit in the FLASH_CR1/2 register to start the bank erase with protection removal operation. Then wait until the QW1/2 bit is cleared in the corresponding FLASH_SR1/2 register. At that point a bank erase operation has erased the whole bank including the sectors containing PCROP-protected and/or secure-only data, and an option byte change has been automatically performed so that all the protections are disabled.

Note: BER1/2 and START1/2 bits can be set together, so above steps 8 and 9 can be merged.

Be aware of the following warnings regarding to above sequence:

- It is not possible to perform the above sequence on one bank while modifying the protection parameters of the other bank.
- No other option bytes than the one indicated above must be changed, and no protection change must be performed in the bank that is not targeted by the bank erase with protection removal request.
- When one or both of the events above occurs, a simple bank erase occurs, no option byte change is performed and no option change error is set.

Flash mass erase sequence

To erase all sectors of both banks simultaneously, excepted for those containing secure-only and protected data, the application software can set the MER bit to 1 in FLASH_OPTCR register, as described below:

1. Check and clear (optional) all the error flags due to previous programming/erase operation. Refer to Section 4.7: FLASH error management for details.
2. Unlock the two FLASH_CR1/2 registers and FLASH_OPTCR register, as described in Section 4.5.1: FLASH configuration protection (only if the registers are not already unlocked).
3. Set the MER bit to 1 in FLASH_OPTCR register. It automatically sets BER1, BER2, START1 and START2 to 1, thus launching a bank erase operation on both banks. Then wait until both QW1 and QW2 bits are cleared in the corresponding FLASH_SR1/2 register.
Flash mass erase with automatic protection-removal sequence

To erase all sectors of both banks simultaneously, including those containing secure-only and protected data, and without performing an RDP regression, proceed as follows:

1. Check and clear (optional) all the error flags due to previous programming/erase operation.
2. Unlock the two FLASH_CR1/2 registers and FLASH_OPTCR register (only if the registers are not already unlocked).
3. If a PCROP-protected area exists, set DMEP1/2 bit in FLASH_PRAR_PRG1/2 register. In addition, program the PCROP area end and start addresses so that the difference is negative. This operation must be performed for both banks.
4. If a secure-only area exists, set DMES1/2 bit in FLASH_SCAR_PRG1/2 register. In addition, program the secure-only area end and start addresses so that the difference is negative. This operation must be performed for both banks.
5. Set all WRPSGn1/2 bits in FLASH_WPSGN_PRG1/2R to 1 to disable all sector write protections. This operation must be performed for both banks.
6. Set the MER bit to 1 in FLASH_OPTCR register, then wait until the QW1/2 bit is cleared in the corresponding FLASH_SR1/2 register. At that point, a Flash bank erase with automatic protection removal is executed on both banks. The sectors containing PCROP-protected and/or secure-only data become unprotected since an option byte change is automatically performed after the mass erase so that all the protections are disabled.

Caution: No other option bytes than the ones mentioned in the above sequence must be changed, otherwise a simple mass erase is executed, no option byte change is performed and no option change error is raised.

4.3.11 FLASH parallel operations

As the non-volatile memory is divided into two independent banks, the embedded Flash memory interface can drive different operations at the same time on each bank. For example a read, write or erase operation can be executed on bank 1 while another read, write or erase operation is executed on bank 2.

In all cases, the sequences described in Section 4.3.8: FLASH read operations, Section 4.3.9: FLASH program operations and Section 4.3.10: FLASH erase operations apply.

4.3.12 Flash memory error protections

Error correction codes (ECC)

The embedded Flash memory supports an error correction code (ECC) mechanism. It is based on the SECDED algorithm in order to correct single errors and detects double errors. This mechanism uses 9 ECC bits per 128-bit Flash word, and applies to user and system memory. For read-only and OTP areas, a stronger 6 ECC bits per 16-bit word is used.

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

More specifically, during each read operation from a 128-bit Flash word, the embedded Flash memory retrieves the 9-bit ECC information, computes the ECC of the Flash word, and compares the result with the reference value. If they do not match, the corresponding
ECC error is raised as described in Section 4.7.6: Error correction code error (SNECCERR/DBECCERR).

During each program operation, a 9-bit ECC code is associated to each 128-bit data Flash word, and the resulting 137-bit Flash word information is written in non-volatile memory.

A similar mechanism applies to read-only and OTP areas.

**Cyclic redundancy codes (CRC)**

The embedded Flash memory implements a cyclic redundancy check (CRC) hardware module. This module checks the integrity of a given user Flash memory area content (see Figure 6: Detailed FLASH architecture).

The area processed by the CRC module can be defined either by sectors or by start/end addresses. It can also be defined as the whole bank (user Flash memory area only).

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

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

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

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

The CRC operation is concurrent with option byte change as the same hardware is used for both operations. To avoid the CRC computation from being corrupted, the application shall complete the option byte change (by reading the result of the change) before running a CRC operation, and vice-versa.

The sequence recommended to configure a CRC operation in the bank 1/2 is the following:

1. Unlock FLASH_CR1/2 register, if not already unlocked.
2. Enable the CRC feature by setting the CRC_EN bit in FLASH_CR1/2.
3. Program the desired data size in the CRC_BURST field of FLASH_CRCCR1/2.
4. Define the user Flash memory area on which the CRC has to be computed. Two solutions are possible:
   - Define the area start and end addresses by programming FLASH_CRCSADD1/2R and FLASH_CRCEADD1/2R, respectively,
   - or select the targeted sectors by setting the CRC_BY_SECT bit in FLASH_CRCCR1/2 and by programming consecutively the target sector numbers in the CRC_SEC field of the FLASH_CRCCR1/2 register. Set ADD_SECT bit after each CRC_SEC programming.
5. Start the CRC operation by setting the START_CRC bit.
6. Wait until the CRC_BUSY1/2 flag is reset in FLASH_SR1/2 register.
7. Retrieve the CRC result in the FLASH_CRCDATAR register.

The CRC can be computed for a whole bank by setting the ALL_BANK bit in the FLASH_CRCCR1/2 register.
Note: The application should avoid running a CRC on PCROP- or secure-only user Flash memory area since it may alter the expected CRC value. A special error flag defined in Section 4.7.9: CRC read error (CRCRDERR) can be used to detect such a case.

CRC computation does not raise standard read error flags such as RDSERR1/2, RDPERR1/2 and DBECCERR1/2. Only CRCRDERR1/2 is raised.

4.3.13 FLASH one-time programmable area

The embedded Flash memory offers a 1024-byte memory area dedicated to application non-confidential one-time programmable data (OTP). This area is composed of 512 words of 16 bits (plus 6 bits of ECC). It cannot be erased and can be written only once. The OTP area can be accessed through the AHB interface from address 0x08FF F000 to 0x08FF F3FC.

OTP data can be programmed by the application software by chunks of 16 bits. Overwriting a 16-bit half-word which has already been programmed leads to ECC errors and is therefore not supported.

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

When reading OTP data with a single error corrected or a double error detected, the embedded Flash memory reports read errors as described in Section 4.7.6: Error correction code error (SNECCERR/DBECCERR).

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

Note: No special read protection mechanism is implemented on the OTP area.

OTP write protection

OTP data are organized as 16 blocks of 32 OTP words, as shown in Table 16. An entire OTP block can be protected (locked) from write accesses by setting the LOCKBi bit corresponding to each OTP block (i = 0 to 15) in the FLASH_OTPBL option byte register. A block can be write-protected whether or not it has been programmed (even partially).

The OTP block locking operation is irreversible and independent from the readout protection level described in Section 4.5.3: Readout protection (RDP). It is not possible to lock a block in RDP level 2.

Note: The OTP area can only be accessed in read mode.

Table 16. Flash memory OTP organization

<table>
<thead>
<tr>
<th>OTP block</th>
<th>AHB address</th>
<th>AHB word</th>
<th>Lock bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 0</td>
<td>0x08FF F000</td>
<td>OTP001</td>
<td>OTP000</td>
</tr>
<tr>
<td></td>
<td>0x08FF F004</td>
<td>OTP003</td>
<td>OTP002</td>
</tr>
<tr>
<td></td>
<td>0x08FF F03C</td>
<td>OTP031</td>
<td>OTP030</td>
</tr>
</tbody>
</table>
 OTP write sequence

Follow the sequence below to write an OTP word:

2. Check the protection status of the target OTP word (see Table 16). The corresponding LOCKBi bit must be cleared to 0.
3. Write two OTP words (32 bits) corresponding to the 4-byte aligned address shown in Table 16. Alternatively, the application software can program separately the 16-bit MSB or 16-bit LSB. In this case the first 16-bit write operation starts immediately without waiting for the second one.
4. Check that QW1 bit in FLASH_S R1 has been raised and wait until it is reset to 0.
5. Optionally, lock the OTP block using LOCKBi to prevent illegal data changes.

Note: Do not write twice an OTP 16-bit word, otherwise an ECC error might be generated.

Writing OTP data at byte level is not supported and generates a bus error.
FLASH_OPTCR and FLASH_SR1 are always used when programming OTP data, regardless of the bank swapping option (enabled or not).
To avoid data corruption, it is important to complete the OTP write process (for example by reading back the OTP value), before starting an option change.

4.3.14 FLASH read-only area

The embedded Flash memory offers a 512-byte memory area to store read-only data. This area is mapped as described in Section 4.3.4: Flash memory architecture and usage. It can be accessed through the AHB system port. This read-only area is protected by a robust ECC scheme, as explained in Section 4.3.12: Flash memory error protections.
The read-only information that can be used by the application software are described in Table 17. This information is programmed by STMicroelectronics.

### Table 17. Read-only public data organization

<table>
<thead>
<tr>
<th>Read-only data name</th>
<th>Address</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique device ID</td>
<td>0x08FF F800</td>
<td>U_ID[31:0]</td>
</tr>
<tr>
<td></td>
<td>0x08FF F804</td>
<td>U_ID[63:32]</td>
</tr>
<tr>
<td></td>
<td>0x08FF F808</td>
<td>U_ID[96:64]</td>
</tr>
<tr>
<td>Flash memory size/</td>
<td>0x08FF F80C</td>
<td>Flash memory size[15:0]</td>
</tr>
<tr>
<td>package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>0x08FF F810 to</td>
<td>Reserved information</td>
</tr>
<tr>
<td></td>
<td>0x08FF F9FF</td>
<td></td>
</tr>
</tbody>
</table>

#### 4.3.15 Flash bank and register swapping (STM32H7A3/7B3 only)

##### Flash bank swapping

The embedded Flash memory bank 1 and bank 2 can be swapped in the memory map accessible through AXI interface. This feature can be used after a firmware upgrade to restart the device on the new firmware. Bank swapping is controlled by the SWAP_BANK bit of the FLASH_OPTCR register.

Table 18 shows the memory map that can be accessed from the embedded Flash memory AXI slave interface, depending on the SWAP_BANK bit configuration.

### Table 18. FLASH AXI interface memory map and the swapping option

<table>
<thead>
<tr>
<th>Flash memory area</th>
<th>Flash memory corresponding bank</th>
<th>Start address</th>
<th>End address</th>
<th>Size (bytes)</th>
<th>Region Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>User main memory</td>
<td>SWAP_BANK=0, SWAP_BANK=1 (1)</td>
<td>0x0800 0000</td>
<td>0x0800 1FFF</td>
<td>8 K</td>
<td>Sector 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0800 2000</td>
<td>0x0800 3FFF</td>
<td>8 K</td>
<td>Sector 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0807 E000</td>
<td>0x0807 FFFF</td>
<td>8 K</td>
<td>Sector 63(2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x080F E000</td>
<td>0x080F FFFF</td>
<td>8 K</td>
<td>Sector 127</td>
</tr>
<tr>
<td></td>
<td>Bank 1</td>
<td>0x0810 0000</td>
<td>0x0810 1FFF</td>
<td>8 K</td>
<td>Sector 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0810 2000</td>
<td>0x0810 3FFF</td>
<td>8 K</td>
<td>Sector 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0817 E000</td>
<td>0x0817 FFFF</td>
<td>8 K</td>
<td>Sector 63(2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bank 2</td>
<td>0x081F E000</td>
<td>0x081F FFFF</td>
<td>8 K</td>
<td>Sector 127</td>
</tr>
</tbody>
</table>
The SWAP_BANK bit in FLASH_OPTCR register is loaded from the SWAP_BANK_OPT option bit only after system reset or POR.

To change the SWAP_BANK bit (for example to apply a new firmware update), respect the sequence below:

1. Unlock OPTLOCK bit, if not already unlocked.
2. Set the new desired SWAP_BANK_OPT value in the FLASH_OPTSR_PRG register.
3. Start the option byte change sequence by setting the OPTSTART bit in the FLASH_OPTCR register.
4. Once the option byte change has completed, FLASH_OPTSR_CUR contains the expected SWAP_BANK_OPT value, but SWAP_BANK bit in FLASH_OPTCR has not yet been modified and the bank swapping is not yet effective.
5. Force a system reset or a POR. When the reset rises up, the bank swapping is effective (SWAP_BANK value updated in FLASH_OPTCR) and the new firmware shall be executed.

**Note:** The SWAP_BANK bit in FLASH_OPTCR is read-only and cannot be modified by the application software.

The SWAP_BANK_OPT option bit in FLASH_OPTSR_PRG can be modified whatever the RDP level (i.e. even in level 2), thus allowing advanced firmware upgrade in any level of readout protection.

---

### Table 18. FLASH AXI interface memory map and the swapping option (continued)

<table>
<thead>
<tr>
<th>Flash memory area</th>
<th>Flash memory corresponding bank</th>
<th>SWAP_BANK=0</th>
<th>SWAP_BANK=1 (1)</th>
<th>Start address</th>
<th>End address</th>
<th>Size (bytes)</th>
<th>Region Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>System memory</td>
<td>Bank 1</td>
<td></td>
<td></td>
<td>0x1FF0 0000</td>
<td>0x1FF0 1FFF</td>
<td>8 K</td>
<td>System 1 Sector 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1FF0 2000</td>
<td>0x1FF0 3FFF</td>
<td>8 K</td>
<td>System 1 Sector 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1FF0 E000</td>
<td>0x1FF0 FFFF</td>
<td>8 K</td>
<td>System 1 Sector 7</td>
</tr>
<tr>
<td></td>
<td>Bank 2</td>
<td></td>
<td></td>
<td>0x1FF1 0000</td>
<td>0x1FF1 1FFF</td>
<td>8 K</td>
<td>System 2 Sector 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1FF1 2000</td>
<td>0x1FF1 3FFF</td>
<td>8 K</td>
<td>System 2 Sector 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1FF1 E000</td>
<td>0x1FF1 FFFF</td>
<td>8 K</td>
<td>System 2 Sector 7</td>
</tr>
</tbody>
</table>

1. User memory bank 2 and bank swapping are supported only on STM32H7A3/7B3 devices.
2. Bank 1 and bank 2 are limited to 64 sectors on STM32H7AxG devices.
Figure 12 gives an overview of the bank swapping sequence.

**Figure 12. Flash bank swapping sequence**

- **Begin**
  - Update new firmware in user Flash bank 1/2
  - Set/unset SWAP_BANK_OPTS bit in FLASH_OPTSR_PRG
  - Write START bit in FLASH_OPTSR_CUR to start The option byte change sequence
  - System reset
  - SWAP_BANK of FLASH_OPTSR_CUR is copied to SWAP_BANK bit in FLASH_OPTSR_PRG

- **End**
  - Execution of new firmware

**FLASH_OPTSR_PRG**
- Option byte programming
- Non-volatile memory

**FLASH_OPTSR_CUR**
- Option byte reload
- Non-volatile memory

**FLASH_OPTSR_PRG**
- System reset option byte reload
- New swap policy effective
- User Flash memory
- SWAP_BANK_16gic
- AXI bus

**Non-volatile memory Bank1**

**Non-volatile memory Bank2**
Configuration and option byte register swapping

The embedded Flash memory bank swapping option controlled by the SWAP_BANK bit also swaps the two sets of configuration and option byte registers, as shown in Table 19. One set of registers is related to bank 1 while the other is related to bank 2. Since some registers are not specific to any particular bank, they are mapped onto two different addresses so that the swapping does not affect the access to these registers.

Table 19. Flash register map vs swapping option

<table>
<thead>
<tr>
<th>Address offset (1)</th>
<th>Register name</th>
<th>Register targeting bank (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SWAP_BANK=0</td>
</tr>
<tr>
<td>0x000 or 0x100</td>
<td>FLASH_ACR</td>
<td>N/A</td>
</tr>
<tr>
<td>0x004</td>
<td>0x104</td>
<td>FLASH_KEYR1</td>
</tr>
<tr>
<td>0x008</td>
<td>0x108</td>
<td>FLASH_OPTKEYR</td>
</tr>
<tr>
<td>0x00C</td>
<td>0x10C</td>
<td>FLASH_CR1</td>
</tr>
<tr>
<td>0x010</td>
<td>0x110</td>
<td>FLASH_SR1</td>
</tr>
<tr>
<td>0x014</td>
<td>0x114</td>
<td>FLASH_CCR1</td>
</tr>
<tr>
<td>0x018 or 0x118</td>
<td></td>
<td>FLASH_OPTCR</td>
</tr>
<tr>
<td>0x01C or 0x11C</td>
<td></td>
<td>FLASH_OPTSR_CUR</td>
</tr>
<tr>
<td>0x020 or 0x120</td>
<td></td>
<td>FLASH_OPTSR_PRG</td>
</tr>
<tr>
<td>0x024 or 0x124</td>
<td></td>
<td>FLASH_OPTCCR</td>
</tr>
<tr>
<td>0x028</td>
<td>0x128</td>
<td>FLASH_PRAR_CUR1</td>
</tr>
<tr>
<td>0x02C</td>
<td>0x12C</td>
<td>FLASH_PRAR_PRG1</td>
</tr>
<tr>
<td>0x030</td>
<td>0x130</td>
<td>FLASH_SCAR_CUR1</td>
</tr>
<tr>
<td>0x034</td>
<td>0x134</td>
<td>FLASH_SCAR_PRG1</td>
</tr>
<tr>
<td>0x038</td>
<td>0x138</td>
<td>FLASH_WPSGN_CUR1</td>
</tr>
<tr>
<td>0x03C</td>
<td>0x13C</td>
<td>FLASH_WPSGN_PRG1</td>
</tr>
<tr>
<td>0x040 or 0x140</td>
<td></td>
<td>FLASH_BOOT_CUR</td>
</tr>
<tr>
<td>0x044 or 0x144</td>
<td></td>
<td>FLASH_BOOT_PRG</td>
</tr>
<tr>
<td>0x050</td>
<td>0x150</td>
<td>FLASH_CRCCR1</td>
</tr>
<tr>
<td>0x054</td>
<td>0x154</td>
<td>FLASH_CRCSADD1R</td>
</tr>
<tr>
<td>0x058</td>
<td>0x158</td>
<td>FLASH_CRCEADD1R</td>
</tr>
<tr>
<td>0x05C</td>
<td>0x15C</td>
<td>FLASH_CRCADDR</td>
</tr>
<tr>
<td>0x060</td>
<td>0x160</td>
<td>FLASH_ECC_FA1R</td>
</tr>
<tr>
<td>0x068</td>
<td></td>
<td>FLASH_OTPBL_PRG</td>
</tr>
<tr>
<td>0x06C</td>
<td></td>
<td>FLASH_OTPBL_PRG</td>
</tr>
</tbody>
</table>

1. As shown above, some registers are not dedicated to a specific bank and can be accessed at two different addresses.
2. Bank swapping is supported only on STM32H7A3/7B3 devices.
4.3.16 FLASH reset and clocks

Reset management

The embedded Flash memory can be reset by a core domain reset, driven by the reset and clock control (RCC). The main effects of this reset are the following:

- All registers, except for option byte registers, are cleared, including read and write latencies. If the bank swapping option is changed, it will be applied.
- Most control registers are automatically protected against write operations. To unprotected them, new unlock sequences must be used as described in Section 4.5.1: FLASH configuration protection.

The embedded Flash memory can be reset by a power-on core domain reset, driven by the reset and clock control (RCC). When the reset falls, all option byte registers are reset. When the reset rises up, the option bytes are loaded, potentially applying new features. During this loading sequence, the device remains under reset and the embedded Flash memory is not accessible.

The Reset signal can have a critical impact on the embedded Flash memory:

- The contents of the Flash memory are not guaranteed if a device reset occurs during a Flash memory write or erase operation.
- If a reset occurs while the option byte modification is ongoing, the old option byte values are kept. When it occurs, a new option byte modification sequence is required to program the new values.

Clock management

The embedded Flash memory uses the microcontroller system clock (sys_ck), here the AXI interface clock.

Depending on the device clock and internal supply voltage, specific read and write latency settings usually need to be set in the Flash access control register (FLASH_ACR), as explained in Section 4.3.8: FLASH read operations and Section 4.3.9: FLASH program operations.

4.4 FLASH option bytes

4.4.1 About option bytes

The embedded Flash memory includes a set of non-volatile option bytes. They are loaded at power-on reset and can be read and modified only through configuration registers.

These option bytes are configured by the end-user depending on the application requirements. Some option bytes might have been initialized by STMicroelectronics during manufacturing stage.

This section documents:

- When option bytes are loaded
- How application software can modify them
- What is the detailed list of option bytes, together with their default factory values (i.e. before the first option byte change).
4.4.2 Option byte loading

There are multiple ways of loading the option bytes into embedded Flash memory:

1. **Power-on wakeup**
   - When the device is first powered, the embedded Flash memory automatically loads all the option bytes. During the option byte loading sequence, the device remains under reset and the embedded Flash memory cannot be accessed.

2. **Wakeup from system Standby**
   - When the core power domain, which contains the embedded Flash memory, is switched from Standby mode to DRun mode, the embedded Flash memory behaves as during a power-on sequence.

3. **Dedicated option byte reloading by the application**
   - When the user application successfully modifies the option byte content through the embedded Flash memory registers, the non-volatile option bytes are programmed and the embedded Flash memory automatically reloads all option bytes to update the option registers.

   **Note:** The option bytes read sequence is enhanced thanks to a specific error correction code. In case of security issue, the option bytes may be loaded with default values (see Section 4.4.3: Option byte modification).

4.4.3 Option byte modification

**Changing user option bytes**

A user option byte change operation can be used to modify the configuration and the protection settings saved in the non-volatile option byte area of memory bank 1.

The embedded Flash memory features two sets of option byte registers:

- The first register set contains the current values of the option bytes. Their names have the _CUR extension. All “_CUR” registers are read-only. Their values are automatically loaded from the non-volatile memory after power-on reset, wakeup from system standby or after an option byte change operation.

- The second register set allows the modification of the option bytes. Their names contain the _PRG extension. All “_PRG” registers can be accessed in read/write mode.

When the OPTLOCK bit in FLASH_OPTCR register is set, modifying the _PRG registers is not possible.

When OPTSTART bit is set to 1, the embedded Flash memory checks if at least one option byte needs to be programmed by comparing the current values (_CUR) with the new ones (_PRG). If this is the case and all the other conditions are met (see *Changing security option bytes*), the embedded Flash memory launches the option byte modification in its non-volatile memory and updates the option byte registers with _CUR extension.

If one of the condition described in *Changing security option bytes* is not respected, the embedded Flash memory sets the OPTCHANGEERR flag to 1 in the FLASH_OPTSR_CUR register and aborts the option byte change operation. In this case, the _PRG registers are not overwritten by current option value. The user application can check what was wrong in their configuration.
Unlocking the option byte modification

After reset, the OPTLOCK bit is set to 1 and the FLASH_OPTCR is locked. As a result, the application software must unlock the option configuration register before attempting to change the option bytes. The FLASH_OPTCR unlock sequence is described in Section 4.5.1: FLASH configuration protection.

Option byte modification sequence

To modify user option bytes, follow the sequence below:

1. Unlock FLASH_OPTCR register as described in Section 4.5.1: FLASH configuration protection, unless the register is already unlocked.
2. Write the desired new option byte values in the corresponding option registers (FLASH_XXX_PRG1/2).
3. Set the option byte start change OPTSTART bit to 1 in the FLASH_OPTCR register.
4. Wait until OPT_BUSY bit is cleared.

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

Changing security option bytes

On top of OPTLOCK bit, there is a second level of protection for security-sensitive option byte fields. Specific rules must be followed to update them:

- **Readout protection (RDP)**
  A detailed description of RDP option bits is given in Section 4.5.3. The following rules must be respected to modify these option bits:
  - When RDP is set to level 2, no changes are allowed (except for the SWAP bit). As a result, if the user application attempts to reduce the RDP level, an option byte change error is raised (OPTCHANGEERR bit in FLASH_OPTSR_CUR register), and all the programmed changes are ignored.
  - When the RDP is set to level 1, requiring a change to level 2 is always allowed. When requiring a regression to level 0, an option byte change error can occur if some of the recommendations provided in this chapter have not been followed.
  - When the RDP is set to level 0, switching to level 1 or level 2 is possible without any restriction.

- **Sector write protection (WRPSGn1/2)**
  These option bytes manage group of sector write protection in FLASH_WPSGN_CUR1/2R registers. They can be changed without any restriction when the RDP protection level is different from level 2.

- **PCROP area size (PROT_AREA_START1/2 and PROT_AREA_END1/2)**
  These option bytes configure the size of the PCROP areas in FLASH_PRAR_CUR1/2 registers. They can be increased without any restriction by the Arm® Cortex®-M7 core. To remove or reduce a PCROP area, an RDP level 1 to 0 regression (see Section 4.5.3) or a bank erase with protection removal (see Section 4.3.10) must be
requested at the same time. DMEP must be set to 1 in either FLASH_PRAR_CUR1/2 or FLASH_PRAR_PRG1/2, otherwise an option byte change error is raised.

- **DMEP1/2**

  When this option bit is set, the content of the corresponding PCROP area is erased during a RDP level 1 to 0 regression (see Section 4.5.3) or a bank erase with protection removal (see Section 4.3.10). It is preserved otherwise.

  There are no restrictions in setting DMEP1/2 bit. Resetting DMEP1/2 bit from 1 to 0 can only be done when an RDP level 1 to 0 regression or a bank erase with protection removal is requested at the same time.

- **Secure access mode (SECURITY)**

  The SECURITY option bit activates the secure access mode described in Section 4.5.5. This option bit can be freely set by the application software if such mode is activated on the device. If at least one PCROP or secure-only area is defined as not null, the only way to deactivate the security option bit (from 1 to 0) is to perform an RDP level 1 to 0 regression, when DMEP1/2 is set to 1 in either FLASH_PRAR_CUR1/2 or FLASH_PRAR_PRG1/2 registers, and DMES1/2 is set to 1 in either FLASH_SCAR_CUR1/2 or FLASH_SCAR_PRG1/2.

  If no valid secure-only area and no valid PCROP area are currently defined, the SECURITY option bit can be freely reset.

  **Note:** It is recommended to have both SEC_AREA_START> SEC_AREA_END and PROT_AREA_START> PROT_AREA_END programmed when deactivating the SECURITY option bit during an RDP level 1 to 0 regression.

- **Secure-only area size (SEC_AREA_START1/2 and SEC_AREA_END1/2)**

  These option bytes configure the size of the secure-only areas in FLASH_SCAR_CUR1/2 registers. They can be changed without any restriction by the user secure application or by the ST secure library running on the device. For user non-secure application, the secure-only area size can be removed by performing a bank erase with protection removal (see Section 4.3.10), or an RDP level 1 to 0 regression when DMES1/2 set to 1 in either FLASH_SCAR_CUR1/2 or FLASH_SCAR_PRG1/2 (otherwise an option byte change error is raised).

- **DMES1/2**

  When this option bit is set, the content of the corresponding secure-only area is erased during an RDP level 1 to 0 regression or a bank erase with protection removal, it is preserved otherwise.

  DMES1/2 bits can be set without any restriction. Resetting DMES1/2 bit from 1 to 0 can only be performed when an RDP level 1 to 0 regression or a bank erase with protection removal is requested at the same time.

### 4.4.4 Option bytes overview

**Table 20** lists all the user option bytes managed through the embedded Flash memory registers, as well as their default values before the first option byte change (default factory value).
<table>
<thead>
<tr>
<th>Register</th>
<th>Bitfield</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>31</td>
</tr>
<tr>
<td>FLASH_OPTSR[31:16]</td>
<td></td>
</tr>
<tr>
<td>SWAP_BANK_OP</td>
<td></td>
</tr>
<tr>
<td>OPTCHANGEERROR</td>
<td></td>
</tr>
<tr>
<td>VDD_HSLVL</td>
<td></td>
</tr>
<tr>
<td>DC_DEFAULT</td>
<td>0</td>
</tr>
<tr>
<td>Flash_OPTSR[15:0]</td>
<td>0</td>
</tr>
<tr>
<td>FLASH_BOOT[31:16]</td>
<td>0</td>
</tr>
<tr>
<td>Boot_ADD1[15:0]</td>
<td>0</td>
</tr>
<tr>
<td>FLASH_BOOT[15:0]</td>
<td>0</td>
</tr>
<tr>
<td>Boot_ADD0[15:0]</td>
<td>0</td>
</tr>
<tr>
<td>FLASH_PRAR_x1[31:16]</td>
<td>0</td>
</tr>
<tr>
<td>DMEP1</td>
<td></td>
</tr>
<tr>
<td>PROT_AREA_END1</td>
<td></td>
</tr>
<tr>
<td>FLASH_PRAR_x1[15:0]</td>
<td>0</td>
</tr>
<tr>
<td>PROT_AREA_START1</td>
<td></td>
</tr>
<tr>
<td>FLASH_PRAR_x2[31:16]</td>
<td>0</td>
</tr>
<tr>
<td>DMEP2</td>
<td></td>
</tr>
<tr>
<td>PROT_AREA_END2</td>
<td></td>
</tr>
<tr>
<td>FLASH_PRAR_x2[15:0]</td>
<td>0</td>
</tr>
<tr>
<td>PROT_AREA_START2</td>
<td></td>
</tr>
</tbody>
</table>
Table 20. Option byte organization\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>Bitfield</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default factory value</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x0FF</td>
<td></td>
</tr>
<tr>
<td><strong>FLASH_SCAR_x1[31:16]</strong></td>
<td></td>
</tr>
<tr>
<td>SEC_AREA_END1</td>
<td></td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td><strong>FLASH_SCAR_x1[15:0]</strong></td>
<td></td>
</tr>
<tr>
<td>SEC_AREA_START1</td>
<td></td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td><strong>FLASH_SCAR_x2[31:16]</strong></td>
<td></td>
</tr>
<tr>
<td>SEC_AREA_END2</td>
<td></td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td><strong>FLASH_SCAR_x2[15:0]</strong></td>
<td></td>
</tr>
<tr>
<td>SEC_AREA_START2</td>
<td></td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td><strong>FLASH_WPSGN_x1[31:16]</strong></td>
<td></td>
</tr>
<tr>
<td>WRPSG_127:124</td>
<td></td>
</tr>
<tr>
<td>WRPSG_123:120</td>
<td></td>
</tr>
<tr>
<td>WRPSG_119:88</td>
<td></td>
</tr>
<tr>
<td>WRPSG_87:64</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Default factory value</td>
<td>1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td><strong>FLASH_WPSGN_x1[15:0]</strong></td>
<td></td>
</tr>
<tr>
<td>WRPSG_63:60</td>
<td></td>
</tr>
<tr>
<td>WRPSG_59:56</td>
<td></td>
</tr>
<tr>
<td>WRPSG_53:32</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Default factory value</td>
<td>1 1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

---

\(^{(1)}\) Option bytes: **SEC\_AREA\_START**, **SEC\_AREA\_END**, **DMES1**, **DMES2**
Table 20. Option byte organization (1) (continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>Bitfield</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLASH_WPSGN_x2[31:16]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
</tr>
<tr>
<td></td>
<td>WRPSG[127:124]</td>
</tr>
<tr>
<td></td>
<td>WRPSG[123:120]</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Default factory value</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>FLASH_WPSGN_x2[15:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td></td>
<td>WRPSG[63:60]</td>
</tr>
<tr>
<td></td>
<td>WRPSG[59:56]</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Default factory value</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>FLASH_OTPBL_CUR[31:16]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>FLASH_OTPBL_CUR[15:0]</td>
<td>LOCKBL</td>
</tr>
<tr>
<td></td>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>FLASH_OTPBL_PRG[31:16]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>FLASH_OTPBL_PRG[15:0]</td>
<td>LOCKBL</td>
</tr>
<tr>
<td></td>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

1. The option bytes corresponding to bank 2 are only valid for STM32H7A3/7B3 devices.
4.4.5 Description of user and system option bytes

Below the list of the general-purpose option bytes that can be used by the application:

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

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

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

- **Reset management**
  - `BOR`: Brownout level option, indicating the supply level threshold that activates/releases the reset (see Section 6.5.2: Brownout reset (BOR))
  - `NRST_SDBY`: generates a reset when entering Standby mode if cleared to 0,
  - `NRST_STOP`: generates a reset when entering Stop or Stop2 mode if cleared to 0.

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

- **Bank swapping (STM32H7A3/7B3 only) (see Section 4.3.15 on page 171)**
  - `SWAP_BANK_OPT`: bank swapping option, set to 1 to swap user sectors and registers after boot.

- **Device options**
  - `VDDIO_HSLV`: enables the configuration of pads below 2.5 V for VDDIO power rail if set to 1.
  - `VDDMMC_HSLV`: enables the configuration of pads below 2.5 V for VDDMMC power rail if set to 1.

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

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

- **Reset management:**
  - BOR: brownout level option disabled (option byte value = 0x00)

- **Device options**
  - No bank swapping (option byte value = 0x0)

- **Device working in the full voltage range with I/O speed optimization at low-voltage disabled (VDDIO_HSLV=VDDMMC_HSLV=0)**
Refer to Section 4.9: FLASH registers for details.

4.4.6 Description of data protection option bytes

Below the list of the option bytes that can be used to enhance data protection:

- **RDP[7:0]:** Readout protection level (see Section 4.5.3 on page 187 for details).
- **WRPSGn1/2:** write protection option of the corresponding group of four consecutive sectors in Bank 1 (respectively Bank 2). It is active low. Refer to Section on page 191 for details.
  - Bit N: Group embedding sectors 4 x N to 4 x N + 3
- **PROT_AREAx:** Proprietary code readout protection (refer to Section 4.5.4 on page 191 for details)
  - PROT_AREA_START1 (respectively PROT_AREA_END1) contains the first (respectively last) 256-byte block of the PCROP zone in Bank 1
  - PROT_AREA_START2 (respectively PROT_AREA_END2) contains the first (respectively last) 256-byte block of the PCROP zone in Bank 2
  - DMEP1/2: when set to 1, the PCROP area in Bank 1 (respectively Bank 2) is erased during a RDP protection level regression (change from level 1 to 0) or a bank erase with protection removal.
- **SEC_AREAx:** secure access only zones definition (refer to Section 4.5.5 on page 192 for details).
  - SEC_AREA_START1 (respectively SEC_AREA_END1) contains the first (respectively last) 256-byte block of the secure access only zone in Bank 1
  - SEC_AREA_START2 (respectively SEC_AREA_END2) contains the first (respectively last) 256-byte block of the secure access only zone in Bank 2
  - DMES1/2: when set to 1 the secure access only zone in Bank 1 (respectively Bank 2) is erased during a RDP protection level regression (change from level 1 to 0), or a bank erase with protection removal.

Secure areas can be set only on STM32H7B0 and STM32H7B3 devices.

- **SECURITY:** this non-volatile option can be used by the application to manage secure access mode, as described in Section 4.5.5.
  This bit cannot be set on STM32H7A3 devices, since the Secure access mode is not supported.

- **ST_RAM_SIZE:** this non-volatile option defines the amount of DTCM RAM root secure services (RSS) can use during execution when the SECURITY bit is set. The DTCM RAM is always fully available for the application whatever the option byte configuration.

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

- RDP level 0 (option byte value = 0xAA)
- Flash bank erase operations do not impact secure-only and PCROP data areas when enabled by the application (DMES1/2=DMEP1/2=0)
- PCROP and secure-only zone protections disabled (start addresses higher than end addresses)
- Write protection enabled (all option byte bits set to 1)
- Secure access mode disabled (SECURITY option byte value = 0)
- RSS can use the full DTCM RAM for executing its services (ST_RAM_SIZE = 11)
Refer to Section 4.9: FLASH registers for details.

4.4.7 Description of boot address option bytes

Below the list of option bytes that can be used to configure the appropriate boot address for your application:
- BOOT_ADD0/1: MSB of the Arm® Cortex®-M7 boot address when BOOT pin is low (respectively high)

When STMicroelectronics delivers the device, the values programmed in the data protection option bytes are the following:
- Arm® Cortex®-M7 boot address (MSB): 0x0800 (BOOT pin low for user Flash memory) and system memory 0x1FF0 (BOOT pin high for System Flash memory)

Refer to Section 4.9: FLASH registers for details.

4.5 FLASH protection mechanisms

Since sensitive information can be stored in the Flash memory, it is important to protect it against unwanted operations such as reading confidential areas, illegal programming of protected area, or illegal Flash memory erasing.

The embedded Flash memory implements the following protection mechanisms that can be used by end-user applications to manage the security of embedded non-volatile storage:
- Configuration protection
- Global device Readout protection (RDP)
- Write protection
- Proprietary code readout protection (PCROP)
- Secure access mode areas
- OTP area locking

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

4.5.1 FLASH configuration protection

The embedded Flash memory uses hardware mechanisms to protect the following assets against unwanted or spurious modifications (e.g. software bugs):
- Option bytes change
- Write operations
- Erase commands
- Interrupt masking

More specifically, write operations to embedded Flash memory control registers (FLASH_CR1/2 and FLASH_OPTCR) are not allowed after reset.

The following sequence must be used to unlock FLASH_CR1/2 register:
1. Program KEY1 to 0x45670123 in FLASH_KEYR1/2 key register.
2. Program KEY2 to 0xCDEF89AB in FLASH_KEYR1/2 key register.
3. LOCK1/2 bit is now cleared and FLASH_CR1/2 is unlocked.

The following sequence must be used to unlock FLASH_OPTCR register:
1. Program OPTKEY1 to 0x08192A3B in FLASH_OPTKEYR option key register.
2. Program OPTKEY2 to 0x4C5D6E7F in FLASH_OPTKEYR option key register.
3. OPTLOCK bit is now cleared and FLASH_OPTCR register is unlocked.

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

The FLASH_CR1/2 (respectively FLASH_OPTCR) register can be locked again by software by setting the LOCK1/2 bit in FLASH_CR1/2 register (respectively OPTLOCK bit in FLASH_OPTCR register).

In addition the FLASH_CR1/2 register remains locked and a bus error is generated when the following operations are executed:

- programming a third key value
- writing to a different register belonging to the same bank than FLASH_KEYR1/2 before FLASH_CR1/2 has been completely unlocked (KEY1 programmed but KEY2 not yet programmed)
- writing less than 32 bits to KEY1 or KEY2.

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

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

The embedded Flash memory configuration registers protection is summarized in Table 21.

### Table 21. Flash interface register protection summary

<table>
<thead>
<tr>
<th>Register name</th>
<th>Unlocking register</th>
<th>Protected asset</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLASH_ACR</td>
<td>N/A</td>
<td>-</td>
</tr>
<tr>
<td>FLASH_KEYR1/2</td>
<td>N/A</td>
<td>-</td>
</tr>
<tr>
<td>FLASH_OPTKEYR</td>
<td>N/A</td>
<td>-</td>
</tr>
<tr>
<td>FLASH_CR1/2</td>
<td>FLASH_KEYR1/2</td>
<td>Write operations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Erase commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interrupt generation masking sources</td>
</tr>
<tr>
<td>FLASH_SR1/2</td>
<td>N/A</td>
<td>-</td>
</tr>
<tr>
<td>FLASH_CCR1/2</td>
<td>N/A</td>
<td>-</td>
</tr>
<tr>
<td>FLASH_OPTCR</td>
<td>FLASH_OPTKEYR</td>
<td>Option bytes change</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mass erase</td>
</tr>
<tr>
<td>FLASH_OPTSR_PRG</td>
<td>FLASH_OPTCR</td>
<td>Option bytes change. See Section 4.4.3: Option byte modification for details.</td>
</tr>
<tr>
<td>FLASH_OPTCCR</td>
<td>N/A</td>
<td>-</td>
</tr>
<tr>
<td>FLASH_PRAR_PRG1/2</td>
<td>FLASH_OPTCR</td>
<td>Option bytes (PCROP). See Section 4.4.3: Option byte modification for details.</td>
</tr>
</tbody>
</table>
The purpose of embedded Flash memory write protection is to protect the embedded Flash memory against unwanted modifications of the non-volatile code and/or data. Any Flash group of four consecutive 8-Kbyte sectors can be independently write-protected or unprotected by clearing/setting the corresponding WRPSGn1/2 bit in the FLASH_WPSGN_CUR1/2R register.

A write-protected group of sectors can neither be erased nor programmed. As a result, a bank erase cannot be performed if one group of sectors is write-protected, unless a bank erase is executed during an RDP level 1 to 0 regression (see Section : Flash bank erase with automatic protection-removal sequence for details).

The embedded Flash memory write-protection user option bits can be modified without any restriction when the RDP level is set to level 0 or level 1. When it is set to level 2, the write protection bitfield can no more be changed in the option bytes.

Note: PCROP or secure-only areas are write and erase protected.

Write protection errors are documented in Section 4.7: FLASH error management.

Table 21. Flash interface register protection summary (continued)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Unlocking register</th>
<th>Protected asset</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLASH_SCAR_PRG1/2</td>
<td>FLASH_OPTCR</td>
<td>Option bytes (security). See Section 4.4.3: Option byte modification for details.</td>
</tr>
<tr>
<td>FLASH_WPSGN_PRG1/2</td>
<td>FLASH_OPTCR</td>
<td>Option bytes (write protection)</td>
</tr>
<tr>
<td>FLASH_BOOT_PRGR</td>
<td>FLASH_OPTCR</td>
<td>Option bytes (boot)</td>
</tr>
<tr>
<td>FLASH_CRCCR1/2</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>FLASH_CRCADD1/2R</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>FLASH_CRCEADD1/2R</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>FLASH_CRCADDR</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>FLASH_ECC_FAI/2R</td>
<td>N/A</td>
<td>-</td>
</tr>
<tr>
<td>FLASH_OTPBL_PRG</td>
<td>N/A</td>
<td>OTP write protection</td>
</tr>
</tbody>
</table>

1. Once set, the OTP Block Lock bits cannot be reset by the application.
4.5.3 Readout protection (RDP)

The embedded Flash memory readout protection is global as it does not apply only to the embedded Flash memory, but also to the other secured regions. This is done by using dedicated security signals.

In this section other secured regions are defined as:
- Backup SRAM
- RTC backup registers
- Encrypted regions protected by on-the-fly decryption engine (OTFDEC)

The global readout protection level is set by writing the values given in Table 22 into the readout protection (RDP) option byte (see Section 4.4.3: Option byte modification).

<table>
<thead>
<tr>
<th>RDP option byte value</th>
<th>Global readout protection level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xAA</td>
<td>Level 0</td>
</tr>
<tr>
<td>0xCC</td>
<td>Level 2</td>
</tr>
<tr>
<td>Any other value</td>
<td>Level 1(^{(1)})</td>
</tr>
</tbody>
</table>

1. Default protection level when RDP option byte is erased.

Definitions of RDP global protection level

RDP Level 0 (no protection)

When the global read protection level 0 is set, all read/program/erase operations from/to the user Flash memory are allowed (if no others protections are set). This is true whatever the boot configuration (boot from user or system Flash memory, boot from RAM), and whether the debugger is connected to the device or not. Accesses to the other secured regions are also allowed.

RDP Level 1 (Flash memory content protection)

When the global read protection level 1 is set, the below properties apply:
- The Flash memory content is protected against debugger and potential malicious code stored in RAM. Hence as soon as any debugger is connected or has been connected, or a boot is configured in embedded RAM (intrusion), the embedded Flash memory prevents any accesses to Flash memory.
- When no intrusion is detected (no boot in RAM, no boot in System Flash memory and no debugger connected), all read/program/erase operations from/to the user Flash memory are allowed (if no others protections are set). Accesses to the other secured regions are also allowed.
- When an intrusion is detected, no accesses to the user Flash memory can be performed. A bus error is generated when a read access is requested to the Flash memory.
memory. In addition, no accesses to other secured regions (read or write) can be performed.

- When performing an RDP level regression, i.e. programming the RDP protection to level 0, the user Flash memory and the other secured regions are erased, as described in RDP protection transitions.
- When booting on STM32 bootloader in standard system memory, only the identification services are available (GET_ID_COMMAND, GET_VER_COMMAND and GET_CMD_COMMAND).
- When booting from STMicroelectronics non-secure bootloader, only the identification services are available (GET_ID_COMMAND, GET_VER_COMMAND and GET_CMD_COMMAND).

**RDP Level 2 (device protection and intrusion prevention)**

When the global read protection level 2 is set, the below rules apply:

- All debugging features are disabled.
- Like level 0, all read/write/erase operations from/to the user Flash memory are allowed since the debugger and the boot from RAM and System Flash memory are disabled. Accesses to the other secured regions are also allowed.
- Booting from RAM is no more allowed.
- The user option bits described in Section 4.4 can no longer be changed except for the SWAP bit.

**Caution:** Memory read protection level 2 is an irreversible operation. When level 2 is activated, the level of protection cannot be changed back to level 0 or level 1.

**Note:** The JTAG port is permanently disabled when level 2 is active (acting as a JTAG fuse). As a consequence, STMicroelectronics is not able to perform analysis on defective parts on which the level 2 protection has been set.

Apply a power-on reset if the global read protection level 2 is set while the debugger is still connected.

The above RDP global protection is summarized in Table 23.

**Table 23. Protection vs RDP Level(1)**

<table>
<thead>
<tr>
<th>Boot area</th>
<th>RDP</th>
<th>Debugger connected</th>
<th>User Flash memory access(2)</th>
<th>System Flash memory access (3)/(4)</th>
<th>Other secured regions(5)</th>
<th>Option Bytes access</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Flash memory</td>
<td>Level 0</td>
<td>Yes(6)/No</td>
<td>R/W/E</td>
<td>R</td>
<td>R/W</td>
<td>R/W</td>
<td></td>
</tr>
<tr>
<td>Level 1</td>
<td>No</td>
<td>No</td>
<td>R/W/E</td>
<td>R</td>
<td>R/W</td>
<td>R/W</td>
<td></td>
</tr>
</tbody>
</table>

---

(1) For more information, refer to Table 23 in the document.
Table 23. Protection vs RDP Level\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Boot area</th>
<th>Inputs</th>
<th>Effects</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM or System Flash memory</td>
<td>RDP</td>
<td>User Flash memory access(^{(2)})</td>
<td>System Flash memory access (^{(3)})/(4)</td>
</tr>
<tr>
<td>Level 0</td>
<td>Yes(^{(3)}/No</td>
<td>R/W/E</td>
<td>R</td>
</tr>
<tr>
<td>Level 1</td>
<td>Yes(^{(3)}/No</td>
<td>illegal access(^{(7)})</td>
<td>R</td>
</tr>
<tr>
<td>Level 2</td>
<td>No</td>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

1. R = read, W = write, E = erase.
2. PCROP (see Section 4.5.4) and secure-only access control (see Section 4.5.5) applies.
3. Read accesses to secure boot and secure libraries stored in system Flash bank 1 possible only from STMicroelectronics code.
4. Write and erase operations to System Flash bank 2 is possible only from STMicroelectronics secure update library.
5. The "other secured regions" are defined at the beginning of this section.
6. JTAG interface disabled while secure libraries are executed.
7. Read protection error (RDPERR) with bus error on read operations, Write protection error (WRPERR) on write/erase operations.

RDP protection transitions

Figure 13 shows how to switch from one RDP level to another. The transition is effective after successfully writing the option bytes including RDP (refer to Section 4.4.3 for details on how to change the option bytes).

Figure 13. RDP protection transition scheme
When the current RDP level is RDP level 1, requesting a new RDP level 0 can cause a full or partial erase:

- The user Flash memory area of the embedded Flash memory is fully or partially erased:
  - A partial sector erase occurs if PCROP (respectively secure-only) areas are preserved by the application. It happens when both DMEP1/2 bits (respectively DMES1/2 bits) are cleared to 0 in FLASH_PRAR_CUR1/2 and FLASH_PRAR_PRG1/2 (respectively FLASH_SCAR_CUR1/2 and FLASH_SCAR_PRG1/2). The sectors belonging to the preserved area(s) are not erased.
  - A full bank erase occurs when at least one DMEP1/2 bit is set to 1 in FLASH_PRAR_CUR1/2 or FLASH_PRAR_PRG1/2, and at least one DMES1/2 bit is set to 1 in FLASH_SCAR_CUR1/2 or FLASH_SCAR_PRG1/2.

Note: Data in write protection area are not preserved during RDP regression.

- The other secured regions are also erased, i.e.:
  - Backup SRAM and RTC backup registers are fully erased.
  - The content stored in encrypted regions protected by on-the-fly decryption engine (OTFDEC) are no more accessible in the clear, as OTFDEC registers are erased.

OTP and read-only regions are not affected by RDP regression and remain unchanged.

During a level regression, if a PCROP area overlaps with a secure-only area, the embedded Flash memory performs the erase operation depending on the DMES/DMEP options bits (see strike-through areas in red in Figure 14). More specifically:

- When DMEP is set in FLASH_PRAR_CUR1/2 or FLASH_PRAR_PRG1/2, the PCROP area is erased (overlapped or not with secure-only area).
- When DMES is set in FLASH_SCAR_CUR1/2 or FLASH_SCAR_PRG1/2, the secure-only area is erased (overlapped or not with PCROP area).

Note: The sector protections (PCROP, secure-only) are removed only if the protected sector boundaries are modified by the user application.

---

### Table 24. RDP transition and its effects

<table>
<thead>
<tr>
<th>RDP transition</th>
<th>RDP option update</th>
<th>Effect on device</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Debug connect</td>
<td>Option bytes change</td>
</tr>
<tr>
<td>Level before change</td>
<td>Level after change</td>
<td></td>
</tr>
<tr>
<td>L0</td>
<td>L1</td>
<td>not 0xAA and not 0xCC</td>
</tr>
<tr>
<td>L0</td>
<td>L2</td>
<td>0xCC</td>
</tr>
<tr>
<td>L1</td>
<td>L2</td>
<td>0xCC</td>
</tr>
<tr>
<td>L0</td>
<td>L0</td>
<td>0xAA</td>
</tr>
<tr>
<td>L1</td>
<td>L1</td>
<td>not 0xAA and not 0xCC</td>
</tr>
</tbody>
</table>

1. Except for bank swapping option bit.
**About RDP protection errors**

Whatever the RDP level, the corresponding error flag is raised when an illegal read or write access is detected (see Section 4.7: FLASH error management).

### 4.5.4 Proprietary code readout protection (PCROP)

The embedded Flash memory allows the definition of an “executable-only” area in the user area of each Flash memory bank. In this area, only instruction fetch transactions from the system, i.e. no data access (data or literal pool) are allowed. This protection is particularly efficient to protect third party software intellectual property.

**Note:** Executable-only area usage requires the native code to be compiled accordingly using “execute-only” option.

#### PCROP area programming

One PCROP area can be defined in bank 1 (respectively bank 2) by setting the PROT_AREA_END1 and PROT_AREA_START1 (respectively PROT_AREA_END2 and PROT_AREA_START2) option bytes so that the END address is strictly higher than the START address. PROT_AREA_START and PROT_AREA_END are defined with a granularity of 256 bytes. This means that the actual PCROP area size (in bytes) is defined by:

\[
(\text{PROT\_AREA\_END} - \text{PROT\_AREA\_START}) + 1 \times 256
\]

As an example, to set a PCROP area on the first 4 Kbytes of user bank 1 (i.e. from address 0x0800 0000 to address 0x0800 0FFF, both included), the embedded Flash memory must be configured as follows:

- PROT\_AREA\_START1[11:0] = 0x000
- PROT\_AREA\_END1[11:0] = 0x00F

The protected area size defined above is equal to:

\[
(\text{PROT\_AREA\_END} - \text{PROT\_AREA\_START}) + 1 \times 256 = 16 \times 256 \text{ bytes} = 4 \text{ Kbytes.}
\]
The minimum execute-only PCROP area that can be set is 32 Flash words (or 512 bytes). The maximum area is the whole user Flash memory, configured by setting to the same value the PCROP area START and END addresses.

**Note:** It is recommended to align PCROP area size with 8 Kbytes Flash sector granularity in order to avoid access right issues.

**PCROP area properties**

Each valid PCROP area has the following properties:

- **Arm® Cortex®-M7 debug events** are ignored while executing code in this area.
- Only the CPU can access it (Master ID filtering), using only instruction fetch transactions. In all other cases, accessing the PCROP area is illegal (see below).
- Illegal transactions to a PCROP area (i.e. data read or write, not fetch) are managed as below:
  - Read operations return a zero, write operations are ignored.
  - No bus error is generated but an error flag is raised (RDPERR for read, WRPERR for write).
- A valid PCROP area is erase-protected. As a result:
  - No erase operations to a sector located in this area is possible (including the sector containing the area start address and the end address)
  - No mass erase can be performed if a single valid PCROP area is defined, except during level regression or erase with protection removal.
- Only the CPU can modify the PCROP area definition and DMEP1/2 bits, as explained in **Changing user option bytes** in Section 4.4.3.
- During an RDP level 1 to 0 regression where the PCROP area is not null
  - The PCROP area content is not erased if the corresponding DMEP1/2 bit are both cleared to 0 in FLASH_PRAR_CUR1/2 and FLASH_PRAR_PRG1/2 registers.
  - The PCROP area content is erased if either of the corresponding DMEP1/2 bit is set to 1 in FLASH_PRAR_CUR1/2 or FLASH_PRAR_PRG1/2 register.

For more information on PCROP protection errors, refer to **Section 4.7: FLASH error management.**

### 4.5.5 Secure access mode (STM32H7B0 and STM32H7B3 only)

The embedded Flash memory allows the definition of a secure-only area in the user area of each Flash memory bank. This area can be accessed only while the CPU executes secure application code. This feature is available only if the SECURITY option bit is set to 1.

Secure-only areas help isolating secure user code from application non-secure code. As an example, they can be used to protect a customer secure firmware upgrade code, a custom secure boot library or a third party secure library.

**Secure-only area programming**

One secure-only area can be defined in bank 1 (respectively bank 2) by setting the SEC_AREA_END1 and SEC_AREA_START1 (respectively SEC_AREA_END2 and SEC_AREA_START2) option bytes so that the END address is strictly higher than the START address. SEC_AREA_START and SEC_AREA_END are defined with a granularity of 256 bytes. This means that the actual secure-only area size (in bytes) is defined by:

\[ ((\text{SEC\\_AREA\\_END} - \text{SEC\\_AREA\\_START}) + 1) \times 256 \]
As an example, to set a secure-only area on the first 8 Kbytes of user bank 2 (i.e. from address 0x0810 0000 to address 0x0810 1FFF, both included), the embedded Flash memory must be configured as follows:

- **SEC_AREA_START2[11:0] = 0x000**
- **SEC_AREA_END2[11:0] = 0x01F**

The secure-only area size defined above is equal to:

\[
\text{[(SEC_AREA_END - SEC_AREA_START) + 1] x 256 = 32 x 256 bytes = 8 Kbytes.}
\]

**Note:** These option bytes can be modified only by the CPU running ST security library or application secure code, except during RDP level regression or erase with protection removal.

The minimum secure-only area that can be set is 32 Flash words (or 512 bytes). The maximum area is the whole user Flash memory bank, configured by setting to the same value the secure-only area START and END addresses.

**Note:** It is recommended to align the secure-only area size with 8 Kbytes Flash sector granularity in order to avoid access right issues.

### Secure access-only area properties

- **Arm® Cortex®-M7 debug events are ignored while executing code in this area.**
- Only the CPU executing ST secure library or user secure application can access it (Master ID filtering). In all other cases, accessing the secure-only area is illegal (see below).
- Illegal transactions to a secure-only area are managed as follows:
  - Data read transactions return zero. Data write transactions are ignored. No bus error is generated but an error flag is raised (RDSERR for read, WRPERR for write).
  - Read instruction transactions generate a bus error and the RDSERR error flag is raised.
- A valid secure-only area is erase-protected. As a result:
  - No erase operations to a sector located in this area are possible (including the sector containing the area start address and the end address), unless the application software is executed from a valid secure-only area.
  - No mass erase can be performed if a single valid secure-only area is defined, except during level regression, erase with protection removal or when the application software is executed from a valid secure-only area.
- Only the CPU can modify the secure-only area definition and DMES1/2 bits, as explained in Changing user option bytes in Section 4.4.3.
- During an RDP level 1 to 0 regression where the secure-only area is not null:
  - the secure-only area content is not erased if the corresponding DMES1/2 bit are both cleared to 0 in FLASH_SCAR_CUR1/2 and FLASH_SCAR_PRG1/2 registers.
  - the secure-only area content is erased if either of the corresponding DMES1/2 bit is set to 1 in FLASH_SCAR_CUR1/2 or FLASH_SCAR_PRG1/2 register.

For more information on secure-only protection errors, refer to Section 4.7: FLASH error management.
4.6 FLASH low-power modes

4.6.1 Introduction

Table 25 summarizes the behavior of embedded Flash memory in STM32 low-power modes. Embedded Flash memory belongs to the core domain.

<table>
<thead>
<tr>
<th>System state</th>
<th>Core domain</th>
<th>Power mode</th>
<th>Allowed if FLASH Busy</th>
<th>Flash power mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>DRun</td>
<td>CRun or CSleep</td>
<td>VOS0/1/2/3</td>
<td>Run</td>
</tr>
<tr>
<td></td>
<td>DStop</td>
<td>CStop</td>
<td>No</td>
<td>Clock gated or Stopped</td>
</tr>
<tr>
<td></td>
<td>DStop2</td>
<td></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Stop</td>
<td>DStop</td>
<td></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DStop2</td>
<td></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Standby</td>
<td>Off</td>
<td></td>
<td>No</td>
<td>Off</td>
</tr>
</tbody>
</table>

When the system state changes or within a given system state, the embedded Flash memory might get a different voltage supply range (VOS) according to the application. The procedure to switch the embedded Flash memory into various power mode (run, clock gated, stopped, off) is described hereafter.

Note: For more information in the microcontroller power states, refer to the Power control section (PWR).

4.6.2 Managing the FLASH domain switching to DStop or Standby

As explain in Table 25, if the embedded Flash memory informs the reset and clock controller (RCC) that it is busy (i.e. BSY1/2, QW1/2, WBNE1/2 is set), the microcontroller cannot switch the core domain to DStop, DStop2 or Standby mode.

Note: CRC_BUSY1/2 is not taken into account.

There are two ways to release the embedded Flash memory:

- Reset the WBNE1/2 busy flag in FLASH_SR1/2 register by any of the following actions:
  a) Complete the write buffer with missing data.
  b) Force the write operation without filling the missing data by activating the FW1/2 bit in FLASH_CR1/2 register. This forces all missing data “high”.
  c) Reset the PG1/2 bit in FLASH_CR1/2 register. This disables the write buffer and consequently lead to the loss of its content.

- Poll QW1/2 busy bits in FLASH_SR1/2 register until they are cleared. This will indicate that all recorded write, erase and option change operations are complete.

The microcontroller can then switch the domain to DStop, DStop2 or Standby mode.
4.7 FLASH error management

4.7.1 Introduction

The embedded Flash memory automatically reports when an error occurs during a read, program or erase operation. A wide range of errors are reported:

- Write protection error (WRPERR)
- Programming sequence error (PGSERR)
- Strobe error (STRBERR)
- Inconsistency error (INCERR)
- Error correction code error (SNECCERR/DBECCERR)
- Read protection error (RDPERR)
- Read secure error (RDSERR)
- CRC read error (CRCRDERR)
- Option byte change error (OPTCHANGEERR)

The application software can individually enable the interrupt for each error, as detailed in Section 4.8: FLASH interrupts.

Note: For some errors, the application software must clear the error flag before attempting a new operation.

Each bank has a dedicated set of error flags in order to identify which bank generated the error. They are available in Flash Status register 1 or 2 (FLASH_SR1/2).

The errors related to bank 2 do not apply to STM32H7B0 devices, since only bank 1 is supported.

4.7.2 Write protection error (WRPERR)

When an illegal erase/program operation is attempted to the non-volatile memory bank 1 (respectively bank 2), the embedded Flash memory sets the write protection error flag WRPERR1 (respectively WRPERR2) in FLASH_SR1 register (respectively FLASH_SR2).

An erase operation is rejected and flagged as illegal if it targets one of the following memory areas:

- A sector belonging to a valid PCROP area (even partially)
- A sector belonging to a valid secure-only area (even partially) except if the application software is executed from a valid secure-only area
- A sector write-locked with WRPSGn
- The read-only/OTP area
An program operation is ignored and flagged as illegal if it targets one of the following memory areas:

- The system Flash memory (bank 2 only) while the device is not executing ST bootloader code
- A user Flash sector belonging to a valid PCROP area while the device is not executing an ST secure library
- A user Flash sector belonging to a valid secure-only area while the device is not executing user secure code or ST secure library
- A user sector write-locked with WRPSTn
- An OTP block, locked with LOCKBL
- A read-only section
- The bank 1 system Flash memory
- The user main Flash memory when RDP level is 1 and a debugger has been detected on the device, or the CPU has not booted from user Flash memory.
- A reserved area

When WRPERR1/2 flag is raised, the operation is rejected and nothing is changed in the corresponding bank. If a write burst operation was ongoing, WRPERR1/2 is raised each time a Flash word write operation is processed by the embedded Flash memory.

**Note:** WRPERR1/2 flag does not block any new erase/program operation.

_Not resetting the write protection error flag (WRPERR1/2) does not generate a PGSERR error._

WRPERR1/2 flag is cleared by setting CLR_WRPERR1/2 bit to 1 in FLASH_CCR1/2 register.

If WRPERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when WRPERR1/2 flag is raised (see Section 4.8: FLASH interrupts for details).

### 4.7.3 Programming sequence error (PGSERR)

When the programming sequence to the bank 1 (respectively bank 2) is incorrect, the embedded Flash memory sets the programming sequence error flag PGSERR1 (respectively PGSERR2) in FLASH_SR1 register (respectively FLASH_SR2).

More specifically, PGSERR1/2 flag is set if one of below conditions is met:

- A write operation is requested but the program enable bit (PG1/2) has not been set in FLASH_CR1/2 register prior to the request.
- The inconsistency error (INCERR1/2) has not been cleared to 0 before requesting a new write operation.

When PGSERR1/2 flag is raised, the current program operation is aborted and nothing is changed in the corresponding bank. The corresponding write data buffer is also flushed. If a write burst operation was ongoing, PGSERR1/2 is raised at the end of the burst.

**Note:** When PGSERR1/2 flag is raised, there is a risk that the last write operation performed by the application has been lost because of the above protection mechanism. Hence it is recommended to generate interrupts on PGSERR and verify in the interrupt handler if the last write operation has been successful by reading back the value in the Flash memory.
The PGSERR1/2 flag also blocks any new program operation. This means that PGSERR1 (respectively 2) must be cleared before starting a new program operation on bank 1 (respectively bank 2).

PGSERR1/2 flag is cleared by setting CLR_PGSERR1/2 bit to 1 in FLASH_CCR1/2 register.

If PGSERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when PGSERR1/2 flag is raised. See Section 4.8: FLASH interrupts for details.

4.7.4 Strobe error (STRBERR)

When the application software writes several times to the same byte in bank 1 (respectively bank 2) write buffer, the embedded Flash memory sets the strobe error flag STRBERR1 (respectively STRBERR2) in FLASH_SR1 register (respectively FLASH_SR2).

When STRBERR1/2 flag is raised, the current program operation is not aborted and new byte data replace the old ones. The application can ignore the error, proceed with the current write operation and request new write operations. If a write burst was ongoing, STRBERR1/2 is raised at the end of the burst.

STRBERR1/2 flag is cleared by setting CLR_STRBERR1/2 bit to 1 in FLASH_CCR1/2 register.

If STRBERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when STRBERR1/2 flag is raised. See Section 4.8: FLASH interrupts for details.

4.7.5 Inconsistency error (INCERR)

When a programming inconsistency to bank 1 (respectively bank 2) is detected, the embedded Flash memory sets the inconsistency error flag INCERR1 (respectively INCERR2) in register FLASH_SR1 (respectively FLASH_SR2).

More specifically, INCERR flag is set when one of the following conditions is met:

- A write operation is attempted before completion of the previous write operation, e.g.
  - The application software starts a write operation to fill the 128-bit write buffer, but sends a new write burst request to a different Flash memory address before the buffer is full.
  - One master starts a write operation, but before the buffer is full, another master starts a new write operation to the same address or to a different address.

Note: INCERR flag must be cleared before starting a new write operation, otherwise a sequence error (PGSERR) is raised.

- A wrap burst request issued by a master overlaps two or more 128-bit Flash-word addresses, i.e. wrap bursts must be done within 128-bit Flash-word address boundaries.

It is recommended to follow the sequence below to avoid losing data when an inconsistency error occurs:
1. Execute a handler routine when INCERR1 or INCERR2 flag is raised.
2. Stop all write requests to embedded Flash memory.
3. Verify that the write operations that have been requested just before the INCERR event have been successful by reading back the programmed values from the memory.
4. Clear the corresponding INCERR1/2 bit.
5. Restart the write operations where they have been interrupted.

INCERR1/2 flag is cleared by setting CLR_INCERR1/2 bit to 1 in FLASH_CCR1/2 register.

If INCERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when INCERR1/2 flag is raised (see Section 4.8: FLASH interrupts for details).

4.7.6 Error correction code error (SNECCERR/DBECCERR)

When a single error correction is detected during a read from bank 1 (respectively bank 2) the embedded Flash memory sets the single error correction flag SNECCERR1 (respectively SNECCERR2) in FLASH_SR1 register (respectively FLASH_SR2).

When two ECC errors are detected during a read to bank 1 (respectively bank 2), the embedded Flash memory sets the double error detection flag DBECCERR1 (respectively DBECCERR2) in FLASH_SR1 register (respectively FLASH_SR2). When SNECCERR1/2 flag is raised, the corrected read data are returned. Hence the application can ignore the error and request new read operations.

If a read burst operation was ongoing, SNECCERR1/2 or DBECCERR1/2 flag is raised each time a new data is sent back to the requester through the AXI interface.

When SNECCERR1/2 or DBECCERR1/2 flag is raised, the address of the Flash word that generated the error is saved in the FLASH_ECC_FA1/2R register. If the address corresponds to a read-only area or to an OTP area, the OTP_FAIL_ECC bit is also set to 1 in the FLASH_ECC_FA1R register. This register is automatically cleared when the associated flag that generated the error is reset.

Note: In case of successive single correction or double detection errors, only the address corresponding to the first error is stored in FLASH_ECC_FA1/2R register.

When DBECCERR1/2 flag is raised, a bus error is generated. In case of successive double error detections, a bus error is generated each time a new data is sent back to the requester through the AXI interface.

Note: It is not mandatory to clear SNECCERR1/2 or DBECCERR1/2 flags before starting a new read operation.

SNECCERR1/2 (respectively DBECCERR1/2) flag is cleared by setting to 1 CLR_SNECCERR1/2 bit (respectively CLR_DBECCERR1/2 bit) in FLASH_CCR1/2 register.

If SNECCERR1/2 (respectively DBECCERR1/2) bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when SNECCERR1/2 (respectively DBECCERR1/2) flag is raised. See Section 4.8: FLASH interrupts for details.

4.7.7 Read protection error (RDPERR)

When a read operation to a PCROP, a secure-only or a RDP protected area is attempted in non-volatile memory bank 1 (respectively bank 2), the embedded Flash memory sets the
read protection error flag RDPERR1 (respectively RDPERR2) in FLASH_SR1 register (respectively FLASH_SR2).

When RDPERR1/2 flag is raised, the current read operation is aborted but the application can request new read operations. If a read burst was ongoing, RDPERR1/2 is raised each time a data is sent back to the requester through the AXI interface.

Note: A bus error is raised if a standard application attempts to execute on a secure-only or a RDP protected area.

RDPERR1/2 flag is cleared by setting CLR_RDPERR1/2 bit to 1 in FLASH_CCR1/2 register.

If RDPERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when RDPERR1/2 flag is raised (see Section 4.8: FLASH interrupts for details).

4.7.8 Read secure error (RDSERR)

When a read operation is attempted to a secure address in bank 1 (respectively bank 2), the embedded Flash memory sets the read secure error flag RDSERR1 (respectively RDSERR2) in FLASH_SR1 register (respectively FLASH_SR2). For more information, refer to Section 4.5.5: Secure access mode (STM32H7B0 and STM32H7B3 only).

When RDSERR1/2 flag is raised, the current read operation is aborted and the application can request new read operations. If a read burst was ongoing, RDSERR1/2 is raised each time a data is sent back to the requester through the AXI interface.

Note: The bus error is raised only if the illegal access is due to an instruction fetch.

RDSERR1/2 flag is cleared by setting CLR_RDSERR1/2 bit to 1 in FLASH_CCR1/2 register.

If RDSERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when RDSERR1/2 flag is raised (see Section 4.8: FLASH interrupts for details).

4.7.9 CRC read error (CRCRDERR)

After a CRC computation in bank 1 (respectively bank 2), the embedded Flash memory sets the CRC read error flag CRCRDERR1 (respectively CRCRDERR2) in FLASH_SR1 register (respectively FLASH_SR2) when one or more address belonging to a protected area was read by the CRC module. A protected area corresponds to a PCROP area (see Section 4.5.4) or to a secure-only area (see Section 4.5.5).

CRCRDERR1/2 flag is raised when CRCEND1/2 bit is set to 1 (end of CRC calculation). In this case, it is likely that the CRC result is wrong since illegal read operations to protected areas return null values.

CRCRDERR1/2 flag is cleared by setting CLR_CRCRDERR1/2 bit to 1 in FLASH_CCR1/2 register.

If CRCRDERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when CRCRDERR1/2 flag is raised together with CRCEND1/2 bit (see Section 4.8: FLASH interrupts for details).
4.7.10 Option byte change error (OPTCHANGEERR)

When the embedded Flash memory finds an error during an option change operation, it aborts the operation and sets the option byte change error flag OPTCHANGEERR in FLASH_OPTSR_CUR register.

OPTCHANGEERR flag is cleared by setting CLR_OPTCHANGEERR bit to 1 in FLASH_OPTCCR register.

If OPTCHANGEERRIE bit in FLASH_OPTCR register is set to 1, an interrupt is generated when OPTCHANGEERR flag is raised (see Section 4.8: FLASH interrupts for details).

It is mandatory to clean the OPTCHANGEERR flag before starting a new option change. If the OPTCHANGEERR flag is not cleared, the OPTSTART bit cannot be set in the FLASH_OPTCR register.

4.7.11 Miscellaneous HardFault errors

The following events generate a bus error on the corresponding bus interface:

- On AXI system bus:
  - accesses to user Flash memory while RDP is set to 1 and an illegal condition is detected (boot from system Flash memory, boot from RAM, or debugger connected)
  - fetching to secure-only user Flash memory without the correct access rights
- On AHB configuration or system bus:
  - wrong key input to FLASH_KEYR1/2 or FLASH_OPTKEYR
  - 8-bit accesses to system AHB interface

4.8 FLASH interrupts

The embedded Flash memory can generate a maskable interrupt to signal the following events on a given bank:

- Read and write errors (see Section 4.7: FLASH error management)
  - Single ECC error correction during read operation
  - Double ECC error detection during read operation
  - Write inconsistency error
  - Bad programming sequence
  - Strobe error during write operations
  - option change operation error
- Security errors (see Section 4.7: FLASH error management)
  - Write protection error
  - Read protection error
  - Read secure error
  - CRC computation on PCROP or secure-only area error
- Miscellaneous events (described below)
  - End of programming
  - CRC computation complete
These multiple sources are combined into a single interrupt signal, flash_it, which is the only interrupt signal from the embedded Flash memory that drives the NVIC (nested vectored interrupt controller).

You can individually enable or disable embedded Flash memory interrupt sources by changing the mask bits in the FLASH_CR1/2 register. Setting the appropriate mask bit to 1 enables the interrupt.

**Note:** Prior to writing, FLASH_CR1/2 register must be unlocked as explained in **Section 4.5.1: FLASH configuration protection**

The interrupts corresponding to bank 2 do not apply to STM32H7B0 devices.

**Table 26** gives a summary of the available embedded Flash memory interrupt features. As mentioned in the table below, some flags need to be cleared before a new operation is triggered.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Clear flag to resume operation</th>
<th>Bus error (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-of-program event</td>
<td>on bank 1</td>
<td>EOP1</td>
<td>EOPIE1</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>on bank 2</td>
<td>EOP2</td>
<td>EOPIE2</td>
<td>N/A</td>
</tr>
<tr>
<td>CRC complete event</td>
<td>on bank 1</td>
<td>CRCEND1</td>
<td>CRCENDIE1</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>on bank 2</td>
<td>CRCEND2</td>
<td>CRCENDIE2</td>
<td>N/A</td>
</tr>
<tr>
<td>Write protection error</td>
<td>on bank 1</td>
<td>WRPERR1</td>
<td>WRPERRIE1</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>on bank 2</td>
<td>WRPERR2</td>
<td>WRPERRIE2</td>
<td>No</td>
</tr>
<tr>
<td>Programming sequence error</td>
<td>on bank 1</td>
<td>PGSERR1</td>
<td>PGSERRIE1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>on bank 2</td>
<td>PGSERR2</td>
<td>PGSERRIE2</td>
<td>No</td>
</tr>
<tr>
<td>Strobe error</td>
<td>on bank 1</td>
<td>STRBERR1</td>
<td>STRBERRIE1</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>on bank 2</td>
<td>STRBERR2</td>
<td>STRBERRIE2</td>
<td>No</td>
</tr>
<tr>
<td>Inconsistency error</td>
<td>on bank 1</td>
<td>INCERR1</td>
<td>INCERRIE1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>on bank 2</td>
<td>INCERR2</td>
<td>INCERRIE2</td>
<td>No</td>
</tr>
<tr>
<td>ECC single error correction event</td>
<td>on bank 1</td>
<td>SNECCERR1</td>
<td>SNECCERRIE1</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>on bank 2</td>
<td>SNECCERR2</td>
<td>SNECCERRIE2</td>
<td>No</td>
</tr>
<tr>
<td>ECC double error detection event</td>
<td>on bank 1</td>
<td>DBECCERR1</td>
<td>DBECCERRIE1</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>on bank 2</td>
<td>DBECCERR2</td>
<td>DBECCERRIE2</td>
<td>Yes</td>
</tr>
<tr>
<td>Read protection error</td>
<td>on bank 1</td>
<td>RDPERR1</td>
<td>RDPERRIE1</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>on bank 2</td>
<td>RDPERR2</td>
<td>RDPERRIE2</td>
<td>No</td>
</tr>
<tr>
<td>Read secure error</td>
<td>on bank 1</td>
<td>RDSERR1</td>
<td>RDSERRIE1</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>on bank 2</td>
<td>RDSERR2</td>
<td>RDSERRIE2</td>
<td>Yes</td>
</tr>
</tbody>
</table>

1. Applies to accesses through AXI bus only
The status of the individual maskable interrupt sources described in Table 26 (except for option byte error) can be read from the FLASH_SR1/2 register. They can be cleared by setting to 1 the adequate bit in FLASH_CCR1/2 register.

**Note:** No unlocking mechanism is required to clear an interrupt.

### End-of-program event

Setting the end-of-operation interrupt enable bit (EOPIE1/2) in the FLASH_CR1/2 register enables the generation of an interrupt at the end of an erase operation, a program operation or an option byte change on bank 1/2. The EOP1/2 bit in the FLASH_SR1/2 register is also set when one of these events occurs.

Setting CLR_EOP1/2 bit to 1 in FLASH_CCR1/2 register clears EOP1/2 flag.

### CRC end of calculation event

Setting the CRC end-of-calculation interrupt enable bit (CRCENDIE1/2) in the FLASH_CR1/2 register enables the generation of an interrupt at the end of a CRC operation on bank 1/2. The CRCEND1/2 bit in the FLASH_SR1/2 register is also set when this event occurs.

Setting CLR_CRCEND1/2 bit to 1 in FLASH_CCR1/2 register clears CRCEND1/2 flag.
4.9 FLASH registers

4.9.1 FLASH access control register (FLASH_ACR)

Address offset: 0x000 or 0x100
Reset value: 0x0000 0013

For more details, refer to Section 4.3.8: FLASH read operations and Section 4.3.9: FLASH program operations.

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

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

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 \texttt{WRHIGHFREQ}: Flash signal delay

These bits are used to control the delay between non-volatile memory signals during programming operations. Application software has to program them to the correct value depending on the embedded Flash memory interface frequency. Please refer to Table 15 for details.

\textbf{Note}: No check is performed to verify that the configuration is correct.

Two \texttt{WRHIGHFREQ} values can be selected for some frequencies.

Bits 3:0 \texttt{LATENCY}: Read latency

These bits are used to control the number of wait states used during read operations on both non-volatile memory banks. The application software has to program them to the correct value depending on the embedded Flash memory interface frequency and voltage conditions.

0000: zero wait state used to read a word from non-volatile memory
0001: one wait state used to read a word from non-volatile memory
0010: two wait states used to read a word from non-volatile memory
... 
0111: six wait states used to read a word from non-volatile memory

\textbf{Note}: No check is performed by hardware to verify that the configuration is correct.

4.9.2 FLASH key register for bank 1 (FLASH_KEYR1)

Address offset: 0x004
Reset value: 0x0000 0000

FLASH_KEYR1 is a write-only register. The following values must be programmed consecutively to unlock FLASH_CR1 register:

1. 1\textsuperscript{st} key = 0x4567 0123
2. 2\textsuperscript{nd} key = 0xCDEF 89AB
### 4.9.3 FLASH option key register (FLASH_OPTKEYR)

Address offset: 0x008 or 0x108  
Reset value: 0x0000 0000  

FLASH_OPTKEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_OPTCR register:

1. 1st key = 0x0819 2A3B  
2. 2nd key = 0x4C5D 6E7F

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>KEY1R: Non-volatile memory bank 1 configuration access unlock key</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
</tr>
<tr>
<td>w w w w w w w w w w w w w w w w</td>
<td></td>
</tr>
</tbody>
</table>

### 4.9.4 FLASH control register for bank 1 (FLASH_CR1)

Address offset: 0x00C  
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>OPTKEYR: FLASH option bytes control access unlock key</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
</tr>
<tr>
<td>w w w w w w w w w w w w w w w w</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>OPTKEYR: FLASH option bytes control access unlock key</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
</tr>
<tr>
<td>w w w w w w w w w w w w w w w w</td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:29  Reserved, must be kept at reset value.

Bit 28  **CRCRDERRIE1**: Bank 1 CRC read error interrupt enable bit
When CRCRDERRIE1 bit is set to 1, an interrupt is generated when a protected area (PCROP or secure-only) has been detected during the last CRC computation on bank 1. CRCRDERRIE1 can be programmed only when LOCK1 is cleared to 0.
0: no interrupt generated when a CRC read error occurs on bank 1
1: interrupt generated when a CRC read error occurs on bank 1

Bit 27  **CRCNDIE1**: Bank 1 CRC end of calculation interrupt enable bit
When CRCNDIE1 bit is set to 1, an interrupt is generated when the CRC computation has completed on bank 1. CRCNDIE1 can be programmed only when LOCK1 is cleared to 0.
0: no interrupt generated when CRC computation complete on bank 1
1: interrupt generated when CRC computation complete on bank 1

Bit 26  **DBECERRIE1**: Bank 1 ECC double detection error interrupt enable bit
When DBECERRIE1 bit is set to 1, an interrupt is generated when an ECC double detection error occurs during a read operation from bank 1. DBECERRIE1 can be programmed only when LOCK1 is cleared to 0.
0: no interrupt generated when an ECC double detection error occurs on bank 1
1: interrupt generated if an ECC double detection error occurs on bank 1

Bit 25  **SNECERRIE1**: Bank 1 ECC single correction error interrupt enable bit
When SNECERRIE1 bit is set to 1, an interrupt is generated when an ECC single correction error occurs during a read operation from bank 1. SNECERRIE1 can be programmed only when LOCK1 is cleared to 0.
0: no interrupt generated when an ECC single correction error occurs on bank 1
1: interrupt generated when an ECC single correction error occurs on bank 1

Bit 24  **RDSERRIE1**: Bank 1 secure error interrupt enable bit
When RDSERRIE1 bit is set to 1, an interrupt is generated when a secure error (access to a secure-only protected address) occurs during a read operation from bank 1. RDSERRIE1 can be programmed only when LOCK1 is cleared to 0.
0: no interrupt generated when a secure error occurs on bank 1
1: an interrupt is generated when a secure error occurs on bank 1

Bit 23  **RDPERRIE1**: Bank 1 read protection error interrupt enable bit
When RDPERRIE1 bit is set to 1, an interrupt is generated when a read protection error occurs (access to an address protected by PCROP or by RDP level 1) during a read operation from bank 1. RDPERRIE1 can be programmed only when LOCK1 is cleared to 0.
0: no interrupt generated when a read protection error occurs on bank 1
1: an interrupt is generated when a read protection error occurs on bank 1

Bit 22  Reserved, must be kept at reset value.

Bit 21  **INCERRIE1**: Bank 1 inconsistency error interrupt enable bit
When INCERRIE1 bit is set to 1, an interrupt is generated when an inconsistency error occurs during a write operation to bank 1. INCERRIE1 can be programmed only when LOCK1 is cleared to 0.
0: no interrupt generated when a inconsistency error occurs on bank 1
1: interrupt generated when a inconsistency error occurs on bank 1

Bit 20  Reserved, must be kept at reset value.
Bit 19  **STRBERRIE1**: Bank 1 strobe error interrupt enable bit

When STRBERRIE1 bit is set to 1, an interrupt is generated when a strobe error occurs (the master programs several times the same byte in the write buffer) during a write operation to bank 1. STRBERRIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated when a strobe error occurs on bank 1
1: interrupt generated when strobe error occurs on bank 1.

Bit 18  **PGSERRIE1**: Bank 1 programming sequence error interrupt enable bit

When PGSERRIE1 bit is set to 1, an interrupt is generated when a sequence error occurs during a program operation to bank 1. PGSERRIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated when a sequence error occurs on bank 1
1: interrupt generated when sequence error occurs on bank 1.

Bit 17  **WRPERRIE1**: Bank 1 write protection error interrupt enable bit

When WRPERRIE1 bit is set to 1, an interrupt is generated when a protection error occurs during a program operation to bank 1. WRPERRIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated when a protection error occurs on bank 1
1: interrupt generated when a protection error occurs on bank 1.

Bit 16  **EOPIE1**: Bank 1 end-of-program interrupt control bit

Setting EOPIE1 bit to 1 enables the generation of an interrupt at the end of a program operation to bank 1. EOPIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated at the end of a program operation to bank 1.
1: interrupt enabled when at the end of a program operation to bank 1.

Bit 15  **CRC_EN**: Bank 1 CRC control bit

Setting CRC_EN bit to 1 enables the CRC calculation on bank 1. CRC_EN does not start CRC calculation but enables CRC configuration through FLASH_CRCCR1 register.

When CRC calculation is performed on bank 1, it can only be disabled by setting CRC_EN bit to 0. Resetting CRC_EN clears CRC configuration and resets the content of FLASH_CRCDATAR register.

Clearing CRC_EN to 0 sets CRCDATA to 0x0.

CRC_EN can be programmed only when LOCK1 is cleared to 0.

Bit 14  Reserved, must be kept at reset value.

Bit 13  Reserved, must be kept at reset value.

Bits 12:6  **SSN1**: Bank 1 sector erase selection number

These bits are used to select the target sector for an erase operation (they are unused otherwise). SSN1 can be programmed only when LOCK1 is cleared to 0.

0x00: Sector 0 of user Flash bank 1 selected
0x01: Sector 1 of user Flash bank 1 selected
...
0x0F: Sector 15 of user Flash bank 1 selected
...
0x3F: Sector 63 of user Flash bank 1 selected
...
0x7F: Sector 127 of user Flash bank 1 selected

*Note: Bank 1 is limited to 16 and 64 sectors on STM32H7B0 and STM32H7A3xG devices, respectively.*
Bit 5 **START1**: Bank 1 erase start control bit  
START1 bit is used to start a sector erase or a bank erase operation. START1 can be programmed only when LOCK1 is cleared to 0.  
The embedded Flash memory resets START1 when the corresponding operation has been acknowledged. The user application cannot access any embedded Flash memory register until the operation is acknowledged.

Bit 4 **FW1**: Bank 1 write forcing control bit  
FW1 forces a write operation even if the write buffer is not full. In this case all bits not written are set to 1 by hardware. FW1 can be programmed only when LOCK1 is cleared to 0.  
The embedded Flash memory resets FW1 when the corresponding operation has been acknowledged.  
**Note**: Using a force-write operation prevents the application from updating later the missing bits with something else than 1, because it is likely that it will lead to permanent ECC error.  
Write forcing is effective only if the write buffer is not empty (in particular, FW1 does not start several write operations when the force-write operations are performed consecutively).

Bit 3 **BER1**: Bank 1 erase request  
Setting BER1 bit to 1 requests a bank erase operation on bank 1 (user Flash memory only). BER1 can be programmed only when LOCK1 is cleared to 0.  
BER1 has a higher priority than SER1: if both are set, the embedded Flash memory executes a bank erase.  
0: bank erase not requested on bank 1  
1: bank erase requested on bank 1  
**Note**: Write protection error is triggered when a bank erase is required and some sectors are protected.

Bit 2 **SER1**: Bank 1 sector erase request  
Setting SER1 bit to 1 requests a sector erase on bank 1. SER1 can be programmed only when LOCK1 is cleared to 0.  
SER1 has a higher priority than SER1: if both bits are set, the embedded Flash memory executes a bank erase.  
0: sector erase not requested on bank 1  
1: sector erase requested on bank 1  
**Note**: Write protection error is triggered when a sector erase is required on a protected sector.

Bit 1 **PG1**: Bank 1 internal buffer control bit  
Setting PG1 bit to 1 enables internal buffer for write operations to bank 1. This allows preparing program operations even if a sector or bank erase is ongoing. PG1 can be programmed only when LOCK1 is cleared to 0. When PG1 is reset, the internal buffer is disabled for write operations to bank 1, and all the data stored in the buffer but not sent to the operation queue are lost.  
0: Internal buffer disabled for write operations to bank 1  
1: Internal buffer enabled for write operations to bank 1

Bit 0 **LOCK1**: Bank 1 configuration lock bit  
This bit locks the FLASH_CR1 register. The correct write sequence to FLASH_KEYR1 register unlocks this bit. If a wrong sequence is executed, or if the unlock sequence to FLASH_KEYR1 is performed twice, this bit remains locked until the next system reset. LOCK1 can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK1 changes from 0 to 1, the other bits of FLASH_CR1 register do not change.  
0: FLASH_CR1 register unlocked  
1: FLASH_CR1 register locked
4.9.5  FLASH status register for bank 1 (FLASH_SR1)

Address offset: 0x010
Reset value: 0x0000 0000

Bits 31:29  Reserved, must be kept at reset value.

Bit 28  **CRCRDERR1**: Bank 1 CRC read error flag
CRCRDERR1 flag is raised when a word is found read protected during a CRC operation on
bank 1. An interrupt is generated if CRCRDIE1 and CRCEND1 are set to 1. Writing 1 to
CLR_CRCRDERR1 bit in FLASH_CCR1 register clears CRCRDERR1.
0: no protected area detected inside address read by CRC on bank 1
1: a protected area has been detected inside address read by CRC on bank 1. CRC result is
very likely incorrect.

*Note: This flag is valid only when CRCEND1 bit is set to 1*

Bit 27  **CRCEND1**: Bank 1 CRC end of calculation flag
CRCEND1 bit is raised when the CRC computation has completed on bank 1. An interrupt is
generated if CRCENDIE1 is set to 1. It is not necessary to reset CRCEND1 before restarting
CRC computation. Writing 1 to CLR_CRCEND1 bit in FLASH_CCR1 register clears
CRCEND1.
0: CRC computation not complete on bank 1
1: CRC computation complete on bank 1

Bit 26  **DBECCERR1**: Bank 1 ECC double detection error flag
DBECCERR1 flag is raised when an ECC double detection error occurs during a read
operation from bank 1. An interrupt is generated if DECCERRIE1 is set to 1. Writing 1 to
CLR_DBECCERR1 bit in FLASH_CCR1 register clears DBECCERR1.
0: no ECC double detection error occurred on bank 1
1: ECC double detection error occurred on bank 1

Bit 25  **SNECCERR1**: Bank 1 single correction error flag
SNECCERR1 flag is raised when an ECC single correction error occurs during a read
operation from bank 1. An interrupt is generated if SNCCERRIE1 is set to 1. Writing 1 to
CLR_SNECCERR1 bit in FLASH_CCR1 register clears SNECCERR1.
0: no ECC single correction error occurs on bank 1
1: ECC single correction error occurs on bank 1
Bit 24 **RDSERR1**: Bank 1 secure error flag

RDSERR1 flag is raised when a read secure error (read access to a secure-only protected word) occurs on bank 1. An interrupt is generated if RDSERRIE1 is set to 1. Writing 1 to CLR_RDSERR1 bit in FLASH_CCR1 register clears RDSERR1.

0: no secure error occurs on bank 1
1: a secure error occurs on bank 1

Bit 23 **RDPERR1**: Bank 1 read protection error flag

RDPERR1 flag is raised when a read protection error (read access to a PCROP-protected or a RDP-protected area) occurs on bank 1. An interrupt is generated if RDPERRIE1 is set to 1. Writing 1 to CLR_RDPERR1 bit in FLASH_CCR1 register clears RDPERR1.

0: no read protection error occurs on bank 1
1: a read protection error occurs on bank 1

Bit 22 Reserved, must be kept at reset value.

Bit 21 **INCERR1**: Bank 1 inconsistency error flag

INCERR1 flag is raised when an inconsistency error occurs on bank 1. An interrupt is generated if INCERRIE1 is set to 1. Writing 1 to CLR_INCERR1 bit in the FLASH_CCR1 register clears INCERR1.

0: no inconsistency error occurs on bank 1
1: a inconsistency error occurs on bank 1

Bit 20 Reserved, must be kept at reset value.

Bit 19 **STRBERR1**: Bank 1 strobe error flag

STRBERR1 flag is raised when a strobe error occurs on bank 1 (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if the STRBERRIE1 bit is set to 1. Writing 1 to CLR_STRBERR1 bit in FLASH_CCR1 register clears STRBERR1.

0: no strobe error occurs on bank 1
1: a strobe error occurs on bank 1

Bit 18 **PGSERR1**: Bank 1 programming sequence error flag

PGSERR1 flag is raised when a sequence error occurs on bank 1. An interrupt is generated if the PGSERRIE1 bit is set to 1. Writing 1 to CLR_PGSERR1 bit in FLASH_CCR1 register clears PGSERR1.

0: no sequence error occurs on bank 1
1: a sequence error occurs on bank 1

Bit 17 **WRPERR1**: Bank 1 write protection error flag

WRPERR1 flag is raised when a protection error occurs during a program operation to bank 1. An interrupt is also generated if the WRPERRIE1 is set to 1. Writing 1 to CLR_WRPERR1 bit in FLASH_CCR1 register clears WRPERR1.

0: no write protection error occurs on bank 1
1: a write protection error occurs on bank 1

Bit 16 **EOP1**: Bank 1 end-of-program flag

EOP1 flag is set when a programming operation to bank 1 completes. An interrupt is generated if the EOPIE1 is set to 1. It is not necessary to reset EOP1 before starting a new operation. EOP1 bit is cleared by writing 1 to CLR_EOP1 bit in FLASH_CCR1 register.

0: no programming operation completed on bank 1
1: a programming operation completed on bank 1

Bits 15:4 Reserved, must be kept at reset value.
Bit 3 **CRC_BUSY1**: Bank 1 CRC busy flag

CRC_BUSY1 flag is set when a CRC calculation is ongoing on bank 1. This bit cannot be forced to 0. The user must wait until the CRC calculation has completed or disable CRC computation on bank 1.

0: no CRC calculation ongoing on bank 1
1: CRC calculation ongoing on bank 1

Bit 2 **QW1**: Bank 1 wait queue flag

QW1 flag is set when a write, erase or option byte change operation is pending in the command queue buffer of bank 1. It is not possible to know what type of programming operation is present in the queue.

This flag is reset by hardware when all write, erase or option byte change operations have been executed and thus removed from the waiting queue(s). This bit cannot be forced to 0. It is reset after a deterministic time if no other operations are requested.

0: no write, erase or option byte change operations waiting in the operation queues of bank 1
1: at least one write, erase or option byte change operation is waiting in the operation queue of bank 1

Bit 1 **WBNE1**: Bank 1 write buffer not empty flag

WBNE1 flag is set when the embedded Flash memory is waiting for new data to complete the write buffer. In this state, the write buffer is not empty. WBNE1 is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the event below:

– the application software forces the write operation using FW1 bit in FLASH_CR1
– the embedded Flash memory detects an error that involves data loss
– the application software has disabled write operations in this bank

This bit cannot be forced to 0. To reset it, clear the write buffer by performing any of the above listed actions, or send the missing data.

0: write buffer of bank 1 empty or full
1: write buffer of bank 1 waiting data to complete

Bit 0 **BSY1**: Bank 1 busy flag

BSY1 flag is set when an effective write, erase or option byte change operation is ongoing on bank 1. It is not possible to know what type of operation is being executed.

BSY1 cannot be forced to 0. It is automatically reset by hardware every time a step in a write, erase or option byte change operation completes.

0: no programming, erase or option byte change operation being executed on bank 1
1: programming, erase or option byte change operation being executed on bank 1
4.9.6  FLASH clear control register for bank 1 (FLASH_CCR1)

Address offset: 0x014

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>Res.</td>
<td>CLR_CCRDERR1</td>
<td>CLR_CRCEND1</td>
<td>CLR_DBECCERR1</td>
<td>CLR_SNECCERR1</td>
<td>CLR_RDSERR1</td>
<td>CLR_RDPERR1</td>
<td>CLR_INCERR1</td>
<td>CLR_STBERR1</td>
<td>CLR_PGSEN1</td>
<td>CLR_WRPERR1</td>
<td>CLR_EOP1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:29  Reserved, must be kept at reset value.

Bit 28  CLR_CCRDERR1: Bank 1 CCRDERR1 flag clear bit
Setting this bit to 1 resets to 0 CCRDERR1 flag in FLASH_SR1 register.

Bit 27  CLR_CRCEND1: Bank 1 CRCEND1 flag clear bit
Setting this bit to 1 resets to 0 CRCEND1 flag in FLASH_SR1 register.

Bit 26  CLR_DBECCERR1: Bank 1 DBECCERR1 flag clear bit
Setting this bit to 1 resets to 0 DBECCERR1 flag in FLASH_SR1 register. If the SNECCERR1 flag of FLASH_SR1 register is cleared to 0, FLASH_ECC_FA1R register is reset to 0 as well.

Bit 25  CLR_SNECCERR1: Bank 1 SNECCERR1 flag clear bit
Setting this bit to 1 resets to 0 SNECCERR1 flag in FLASH_SR1 register. If the DBECCERR1 flag of FLASH_SR1 register is cleared to 0, FLASH_ECC_FA1R register is reset to 0 as well.

Bit 24  CLR_RDSERR1: Bank 1 RDSERR1 flag clear bit
Setting this bit to 1 resets to 0 RDSERR1 flag in FLASH_SR1 register.

Bit 23  CLR_RDPERR1: Bank 1 RDPERR1 flag clear bit
Setting this bit to 1 resets to 0 RDPERR1 flag in FLASH_SR1 register.

Bit 22  Reserved, must be kept at reset value.

Bit 21  CLR_INCERR1: Bank 1 INCERR1 flag clear bit
Setting this bit to 1 resets to 0 INCERR1 flag in FLASH_SR1 register.

Bit 20  Reserved, must be kept at reset value.

Bit 19  CLR_STBERR1: Bank 1 STRBERR1 flag clear bit
Setting this bit to 1 resets to 0 STRBERR1 flag in FLASH_SR1 register.

Bit 18  CLR_PGSEN1: Bank 1 PGSERR1 flag clear bit
Setting this bit to 1 resets to 0 PGSERR1 flag in FLASH_SR1 register.
4.9.7 **FLASH option control register (FLASH_OPTCR)**

Address offset: 0x018 or 0x118

Reset value: 0x000 0001

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>SWAP_BANK: Bank swapping option configuration bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: bank 1 and bank 2 not swapped</td>
<td></td>
</tr>
<tr>
<td>1: bank 1 and bank 2 swapped</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>OPTCHANGEERRIE: Option byte change error interrupt enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: no interrupt is generated when an error occurs during an option byte change</td>
<td></td>
</tr>
<tr>
<td>1: an interrupt is generated when an error occurs during an option byte change</td>
<td></td>
</tr>
</tbody>
</table>

Bits 29: Reserved, must be kept at reset value.

<table>
<thead>
<tr>
<th>Bit 25</th>
<th>PG_OTP: OTP program control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting PG_OTP bit enables write operations to the OTP area. This bit can be programmed only when OPTLOCK bit is cleared to 0.</td>
<td></td>
</tr>
<tr>
<td>No error sequence and no inconsistency error are generated during OTP write operations. Write operations do not start if PG_OTP is cleared to 0.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>MER: mass erase request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting this bit launches a non-volatile memory bank erase on both banks (i.e. mass erase).</td>
<td></td>
</tr>
<tr>
<td>FLASH_OPTCR, FLASH_CR1 and FLASH_CR2 must be unlocked prior to setting MER high.</td>
<td></td>
</tr>
</tbody>
</table>
Bits 3:2  Reserved, must be kept at reset value.

Bit 1  **OPTSTART**: Option byte start change option configuration bit

OPTSTART triggers an option byte change operation. The user can set OPTSTART only when the OPTLOCK bit is cleared to 0. The embedded Flash memory resets OPTSTART when the option byte change operation has been acknowledged.

The user application cannot modify any embedded Flash memory register until the option change operation has been completed.

Before setting this bit, the user has to write the required values in the FLASH_XXX_PRG registers. The FLASH_XXX_PRG registers will be locked until the option byte change operation has been executed in non-volatile memory.

It is not possible to start an option byte change operation if a CRC calculation is ongoing on bank 1 or bank 2. Trying to set OPTSTART when CRC_BUSY1/2 of FLASH_SR1/2 register is set has not effect; the option byte change does not start and no error is generated.

Bit 0  **OPTLOCK**: FLASH_OPTCR lock option configuration bit

The OPTLOCK bit locks the FLASH_OPTCR register as well as all _PRG registers. The correct write sequence to FLASH_OPTKEYR register unlocks this bit. If a wrong sequence is executed, or the unlock sequence to FLASH_OPTKEYR is performed twice, this bit remains locked until next system reset.

It is possible to set OPTLOCK by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When OPTLOCK changes from 0 to 1, the others bits of FLASH_OPTCR register do not change.

0: FLASH_OPTCR register unlocked
1: FLASH_OPTCR register locked.

### 4.9.8  FLASH option status register (FLASH_OPTSR_CUR)

Address offset: 0x01C or 0x11C

Reset value: 0xXXXX XXXX (see Table 20: Option byte organization)

This read-only register reflects the current values of corresponding option bits.
Bit 31 **SWAP_BANK_OPT**: Bank swapping option status bit

SWAP_BANK_OPT reflects whether bank 1 and bank 2 are swapped or not.

- 0: bank 1 and bank 2 not swapped
- 1: bank 1 and bank 2 swapped

*Note*: The Flash bank swapping is not available on STM32H7B0 devices. The SWAP_BANK option bit must be kept at '0'.

Bit 30 **OPTCHANGEERR**: Option byte change error flag

OPTCHANGEERR flag indicates that an error occurred during an option byte change operation. When OPTCHANGEERR is set to 1, the option byte change operation did not successfully complete. An interrupt is generated when this flag is raised if the OPTCHANGEERRIE bit of FLASH_OPTCR register is set to 1.

Writing 1 to CLR_OPTCHANGEERR of register FLASH_OPTCCR clears OPTCHANGEERR.

- 0: no option byte change errors occurred
- 1: one or more errors occurred during an option byte change operation.

*Note*: The OPTSTART bit in FLASH_OPTCR cannot be set while OPTCHANGEERR is set.

Bit 29 **VDDIO_HSLV**: VDD I/O high-speed at low-voltage status bit

This bit indicates that the VDDIO power rail operates below 2.5 V.

- 0: Product working in the full voltage range, I/O speed optimization at low-voltage disabled
- 1: VDD I/O below 2.5 V, I/O speed optimization at low-voltage feature allowed

Bits 28:26 Reserved, must be kept at reset value.

Bits 25: 22 Reserved, must be kept at reset value.

Bit 21 **SECURITY**: Security enable option status bit

- 0: Security feature disabled
- 1: Security feature enabled.

Bits 20:19 **ST_RAM_SIZE[1:0]**: ST RAM size option status

- 00: 2 Kbytes reserved to ST code
- 01: 4 Kbytes reserved to ST code
- 10: 8 Kbytes reserved to ST code
- 11: 16 Kbytes reserved to ST code

*Note*: This bitfield is effective only when the security is enabled (SECURITY = 1).

   The whole DTCM RAM is always available for the application whatever ST_RAM_SIZE option byte configuration.

Bit 18 **IWDG_FZ_SDBY**: IWDG Standby mode freeze option status bit

When set the independent watchdog IWDG is frozen in system Standby mode.

- 0: Independent watchdog frozen in Standby mode
- 1: Independent watchdog keep running in Standby mode.

Bit 17 **IWDG_FZ_STOP**: IWDG Stop mode freeze option status bit

When set the independent watchdog IWDG is in system Stop mode.

- 0: Independent watchdog frozen in system Stop mode
- 1: Independent watchdog keep running in system Stop mode.
Bit 16 **VDDMMC_HSLV**: VDDMMC I/O high-speed at low-voltage status bit (BELOW_25V)
This bit indicates that the VDDMMC power rail operates below 2.5 V.
0: I/O speed optimization at low-voltage disabled
1: VDDMMC power rail operating below 2.5 V, I/O speed optimization at low-voltage feature allowed

Bits 15:8 **RDP**: Readout protection level option status byte
0xAA: global readout protection level 0
0xCC: global readout protection level 2
others values: global readout protection level 1.

Bit 7 **NRST_STDY**: Core domain Standby entry reset option status bit
0: a reset is generated when entering Standby mode on core domain
1: no reset generated when entering Standby mode on core domain.

Bit 6 **NRST_STOP**: Core domain DStop entry reset option status bit
0: a reset is generated when entering DStop or DStop2 mode on core domain
1: no reset generated when entering DStop or DStop2 mode on core domain.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IWDG_SW**: IWDG control mode option status bit
0: IWDG watchdog is controlled by hardware
1: IWDG watchdog is controlled by software

Bits 3:2 **BOR_LEV**: Brownout level option status bit
These bits reflects the power level that generates a system reset.
00: BOR OFF
01: BOR Level 1, the threshold level is low (around 2.1 V)
10: BOR Level 2, the threshold level is medium (around 2.4 V)
11: BOR Level 3, the threshold level is high (around 2.7 V)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **OPT_BUSY**: Option byte change ongoing flag
OPT_BUSY indicates if an option byte change is ongoing. When this bit is set to 1, the embedded Flash memory is performing an option change and it is not possible to modify any embedded Flash memory register.
0: no option byte change ongoing
1: an option byte change ongoing and all write accesses to Flash registers are blocked until the option byte change completes.
4.9.9 **FLASH option status register (FLASH_OPTSR_PRG)**

Address offset: 0x020 or 0x120

Reset value: 0xXXXX XXXX (see *Table 20: Option byte organization*)

This register is used to program values in corresponding option bits. Values after reset reflects the current values of the corresponding option bits.

![Register Layout](image)

**Bit 31** **SWAP_BANK_OPT**: Bank swapping option configuration bit

SWAP_BANK_OPT option bit is used to configure whether the bank 1 and bank 2 are swapped or not. This bit is loaded with the SWAP_BANK_OPT bit of FLASH_OPTSR_CUR register after a reset.

- 0: bank 1 and bank 2 not swapped
- 1: bank 1 and bank 2 swapped

*Note*: The Flash bank swapping is not available on STM32H7B0 devices. The SWAP_BANK option bit must be kept at '0'.

**Bit 30** Reserved, must be kept at reset value.

**Bit 29** **VDDIO_HSLV**: VDD I/O high-speed at low-voltage configuration bit

This bit indicates that the VDD power rail operates below 2.5 V.

- 0: Product working in the full voltage range, I/O speed optimization at low-voltage disabled
- 1: VDD I/O below 2.5 V, I/O speed optimization at low-voltage feature allowed

**Bits 28:26** Reserved, must be kept at reset value.

**Bits 25:22** Reserved, must be kept at reset value.

**Bit 21** **SECURITY**: Security enable option configuration bit

The SECURITY option bit enables the secure access mode of the device during an option byte change. The change will be taken into account at next reset (next boot sequence). Once it is enabled, the security feature can be disabled if no areas are protected by PCROP or Secure access mode. If there are secure-only or PCROP protected areas, perform a level regression (from level 1 to 0) and set all the bits to unprotect secure-only areas and PCROP areas.

- 0: Security feature disabled
- 1: Security feature enabled.
Bits 20:19 **ST_RAM_SIZE[1:0]**: ST RAM size option configuration bits
- 00: 2 Kbytes reserved to ST code
- 01: 4 Kbytes reserved to ST code
- 10: 8 Kbytes reserved to ST code
- 11: 16 Kbytes reserved to ST code

*Note*: This bitfield is effective only when the security is enabled (SECURITY = 1).

The whole DTCM RAM is always available for the application whatever ST_RAM_SIZE option byte configuration.

Bit 18 **IWDG_FZ_SDBY**: IWDG Standby mode freeze option configuration bit
This option bit is used to freeze or not the independent watchdog IWDG in system Standby mode.
- 0: Independent watchdog frozen in Standby mode
- 1: Independent watchdog keep running in Standby mode.

Bit 17 **IWDG_FZ_STOP**: IWDG Stop mode freeze option configuration bit
This option bit is used to freeze or not the independent watchdog IWDG in system Stop mode.
- 0: Independent watchdog frozen in system Stop mode
- 1: Independent watchdog keep running in system Stop mode.

Bit 16 **VDDMMC_HSLV**: VDDMMC I/O high-speed at low-voltage configuration bit (BELOW_25V)
This bit indicates that the VDDMMC power rail operates below 2.5 V.
- 0: I/O speed optimization at low-voltage disabled
- 1: VDDMMC power rail operating below 2.5 V, I/O speed optimization at low-voltage feature allowed

Bits 15:8 **RDP**: Readout protection level option configuration bits
RDP bits are used to change the readout protection level. This change is possible only when the current protection level is different from level 2. The possible configurations are:
- 0xAA: global readout protection level 0
- 0xCC: global readout protection level 2
- others values: global readout protection level 1.

Bit 7 **NRST_STDY**: Core domain Standby entry reset option configuration bit
- 0: a reset is generated when entering Standby mode on core domain.
- 1: no reset generated when entering Standby mode on core domain.

Bit 6 **NRST_STOP**: Core domain DStop entry reset option configuration bit
- 0: a reset is generated when entering DStop or DStop2 mode on core domain.
- 1: no reset generated when entering DStop or DStop2 mode on core domain.

Bit 5 Reserved, must be kept at reset value.
4.9.10 FLASH option clear control register (FLASH_OPTCCR)

Address offset: 0x024 or 0x124
Reset value: 0x0000 0000

Bit 31 Reserved, must be kept at reset value.

Bit 30 CLR_OPTCHANGEERR: OPTCHANGEERR reset bit
This bit is used to reset the OPTCHANGEERR flag in FLASH_OPTSR_CUR register.
FLASH_OPTCCR is write-only.
It is reset by programming it to 1.

Bits 29:0 Reserved, must be kept at reset value.
4.9.11  FLASH protection address for bank 1 (FLASH_PRAR_CUR1)

Address offset: 0x028
Reset value: 0xXXXX 0XXX (see Table 20: Option byte organization)
This read-only register reflects the current values of corresponding option bits.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMEP1</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td></td>
<td>PROT_AREA_END1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td></td>
<td>PROT_AREA_START1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  DMEP1: Bank 1 PCROP protected erase enable option status bit
If DMEP1 is set to 1, the PCROP protected area in bank 1 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28  Reserved, must be kept at reset value.

Bits 27:16  PROT_AREA_END1: Bank 1 PCROP area end status bits
These bits contain the last 256-byte block of the PCROP area in bank 1.
If this address is equal to PROT_AREA_START1, the whole bank 1 is PCROP protected.
If this address is lower than PROT_AREA_START1, no protection is set on bank 1.

Bits 15:12  Reserved, must be kept at reset value.

Bits 11:0  PROT_AREA_START1: Bank 1 PCROP area start status bits
These bits contain the first 256-byte block of the PCROP area in bank 1.
If this address is equal to PROT_AREA_END1, the whole bank 1 is PCROP protected.
If this address is higher than PROT_AREA_END1, no protection is set on bank 1.

4.9.12  FLASH protection address for bank 1 (FLASH_PRAR_PRG1)

Address offset: 0x02C
Reset value: 0xXXXX 0XXX (see Table 20: Option byte organization)
This register is used to program values in corresponding option bits.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMEP1</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td></td>
<td>PROT_AREA_END1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td></td>
<td>PROT_AREA_START1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  DMEP1: Bank 1 PCROP protected erase enable option status bit
If DMEP1 is set to 1, the PCROP protected area in bank 1 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28  Reserved, must be kept at reset value.

Bits 27:16  PROT_AREA_END1: Bank 1 PCROP area end status bits
These bits contain the last 256-byte block of the PCROP area in bank 1.
If this address is equal to PROT_AREA_START1, the whole bank 1 is PCROP protected.
If this address is lower than PROT_AREA_START1, no protection is set on bank 1.

Bits 15:12  Reserved, must be kept at reset value.

Bits 11:0  PROT_AREA_START1: Bank 1 PCROP area start status bits
These bits contain the first 256-byte block of the PCROP area in bank 1.
If this address is equal to PROT_AREA_END1, the whole bank 1 is PCROP protected.
If this address is higher than PROT_AREA_END1, no protection is set on bank 1.
Bit 31 **DMEP1**: Bank 1 PCROP protected erase enable option configuration bit

If DMEP1 is set to 1, the PCROP protected area in bank 1 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **PROT_AREA_END1**: Bank 1 PCROP area end configuration bits

These bits contain the last 256-byte block of the PCROP area in bank 1.

If this address is equal to PROT_AREA_START1, the whole bank 1 is PCROP protected.

If this address is lower than PROT_AREA_START1, no protection is set on bank 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **PROT_AREA_START1**: Bank 1 PCROP area start configuration bits

These bits contain the first 256-byte block of the PCROP area in bank 1.

If this address is equal to PROT_AREA_END1, the whole bank 1 is PCROP protected.

If this address is higher than PROT_AREA_END1, no protection is set on bank 1.

4.9.13 **FLASH secure address for bank 1 (FLASH_SCAR_CUR1)**

Address offset: 0x030
Reset value: 0xXXXX 0XXX (see Table 20: Option byte organization)

This read-only register reflects the current values of corresponding option bits.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMES</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>SEC_AREA_END1</td>
<td>SEC_AREA_START1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **DMES1**: Bank 1 secure access protected erase enable option status bit

If DMES1 is set to 1, the secure access only area in bank 1 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **SEC_AREA_END1**: Bank 1 secure-only area end status bits

These bits contain the last 256-byte block of the secure-only area in bank 1.

If this address is equal to SEC_AREA_START1, the whole bank 1 is secure access only.

If this address is lower than SEC_AREA_START1, no protection is set on bank 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **SEC_AREA_START1**: Bank 1 secure-only area start status bits

These bits contain the first 256 bytes of block of the secure-only area in bank 1.

If this address is equal to SEC_AREA_END1, the whole bank 1 is secure access only.

If this address is higher than SEC_AREA_END1, no protection is set on bank 1.
4.9.14 FLASH secure address for bank 1 (FLASH_SCAR_PRG1)

Address offset: 0x034

Reset value: 0xXXXX 0XXX (see Table 20: Option byte organization)

This register is used to program values in corresponding option bits.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMES1</td>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td>SEC_AREA_END1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 31 **DMES1**: Bank 1 secure access protected erase enable option configuration bit
If DMES1 is set to 1, the secure access only area in bank 1 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **SEC_AREA_END1**: Bank 1 secure-only area end configuration bits
These bits contain the last block of 256 bytes of the secure-only area in bank 1.
If this address is equal to SEC_AREA_START1, the whole bank 1 is secure access only.
If this address is lower than SEC_AREA_START1, no protection is set on bank 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **SEC_AREA_START1**: Bank 1 secure-only area start configuration bits
These bits contain the first block of 256 bytes of the secure-only area in bank 1.
If this address is equal to SEC_AREA_END1, the whole bank 1 is secure access only.
If this address is higher than SEC_AREA_END1, no protection is set on bank 1.

4.9.15 FLASH write sector group protection for bank 1 (FLASH_WPSGN_CUR1R)

Address offset: 0x038

Reset value: 0xXXXX XXXX

This read-only register reflects the current values of corresponding option bits.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRPSGn1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRPSGn1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
4.9.16  **FLASH write sector group protection for bank 1**

(FLASH_WPSGN_PRG1R)

Address offset: 0x03C

Reset value: 0xXXXX XXXX

This register is used to program values in corresponding option bits.

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

Bits 31:0  **WRPSGn1**: Bank 1 sector group protection option status byte

Each FLASH_WPSGN_CUR1R bit reflects the write protection status of the corresponding group of four consecutive sectors in bank 1 (0: the group is write protected; 1: the group is not write protected)

- Bit 0: Group embedding sectors 0 to 3
- Bit 1: Group embedding sectors 4 to 7
- Bit N: Group embedding sectors 4 x N to 4 x N + 3
- Bit 31: Group embedding sectors 124 to 127

*Note: Bank 1 is limited to 16 and 64 sectors on STM32H7B0 and STM32H7A3xG devices, respectively.*

4.9.17  **FLASH register boot address FLASH_BOOT_CURR)**

Address offset: 0x040 or 0x140

Reset value: 0xXXXX XXXX (see Table 20: Option byte organization)

This register reflects the current values of corresponding option bits.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
4.9.18 FLASH register boot address FLASH_BOOT_PRGR)

Address offset: 0x044 or 0x144

Reset value: 0xXXXX XXXX (see Table 20: Option byte organization)

This register is used to program values in corresponding option bits.

Bits 31:16 **BOOT_ADD1**: Arm® Cortex®-M7 boot address 1
These bits reflect the MSB of the Arm® Cortex®-M7 boot address when the BOOT pin is high.

Bits 15:0 **BOOT_ADD0**: Arm® Cortex®-M7 boot address 0
These bits reflect the MSB of the Arm® Cortex®-M7 boot address when the BOOT pin is low.
4.9.19  FLASH CRC control register for bank 1 (FLASH_CRCCR1)

Address offset: 0x050
Reset value: 0x001C 0000

This register can be modified only if CRC_EN bit is set to 1 in FLASH_CR1 register.

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

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

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

Bit 22  **ALL_BANK:** Bank 1 CRC select bit
When **ALL_BANK** is set to 1, all bank 1 user sectors are added to list of sectors on which the CRC is calculated.

Bits 21:20  **CRC_BURST:** Bank 1 CRC burst size
CRC_BURST bits set the size of the bursts that are generated by the CRC calculation unit.
- 00: every burst has a size of 4 Flash words (128-bit)
- 01: every burst has a size of 16 Flash words (128-bit)
- 10: every burst has a size of 64 Flash words (128-bit)
- 11: every burst has a size of 256 Flash words (128-bit)

Bits 19:18  Reserved, must be kept at reset value.

Bit 17  **CLEAN_CRC:** Bank 1 CRC clear bit
Setting **CLEAN_CRC** to 1 clears the current CRC result stored in the FLASH_CRCDATAR register.

Bit 16  **START_CRC:** Bank 1 CRC start bit
**START_CRC** bit triggers a CRC calculation on bank 1 using the current configuration. No CRC calculation can launched when an option byte change operation is ongoing because all write accesses to embedded Flash memory registers are put on hold until the option byte change operation has completed.

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

Bit 10  **CLEAN_SECT:** Bank 1 CRC sector list clear bit
Setting **CLEAN_SECT** to 1 clears the list of sectors on which the CRC is calculated.

Bit 9  **ADD_SECT:** Bank 1 CRC sector select bit
Setting **ADD_SECT** to 1 adds the sector whose number is **CRC_SECT** to the list of sectors on which the CRC is calculated.
Bit 8  **CRC_BY_SECT**: Bank 1 CRC sector mode select bit
When CRC_BY_SECT is set to 1, the CRC calculation is performed at sector level, on the sectors present in the list of sectors. To add a sector to this list, use ADD_SECT and CRC_SECT bits. To clean the list, use CLEAN_SECT bit.
When CRC_BY_SECT is reset to 0, the CRC calculation is performed on all addresses between CRC_START_ADDR and CRC_END_ADDR.

Bit 7  Reserved, must be kept at reset value.

Bits 6:0  **CRC_SECT**: Bank 1 CRC sector number
CRC_SECT is used to select one user Flash sectors to be added to the list of sectors on which the CRC is calculated. The CRC can be computed either between two addresses (using registers FLASH_C RCSADD1R and FLASH_CRCEADD1R) or on a list of sectors using this register. If this latter option is selected, it is possible to add a sector to the list of sectors by programming the sector number in CRC_SECT and then setting to 1 ADD_SECT. The list of sectors can be erased either by setting CLEAN_SECT bit or by disabling the CRC computation. CRC_SECT can be set only when CRC_EN of FLASH_CR register is set to 1.

0b0000000: sector 0 of bank 1 for CRC
0b0000001: sector 1 of bank 1 for CRC
...
0b0001111: sector 15 of bank 1 for CRC
...
0b0111111: sector 63 of bank 1 for CRC
...
0b1111111: sector 127 of bank 1 for CRC

**Note**: Bank 1 is limited to 16 and 64 sectors on STM32H7B0 and STM32H7AxG devices, respectively.

### 4.9.20  **FLASH CRC start address register for bank 1**
**(FLASH_C RCSADD1R)**

**Address offset**: 0x054

**Reset value**: 0x0000 0000

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

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

**Bits 31:20  Reserved, must be kept at reset value.**

**Bits 19:2**  **CRC_START_ADDR[19:2]**: CRC start address on bank 1
CRC_START_ADDR is used when CRC_BY_SECT is cleared to 0. It must be programmed to the start address of the bank 1 memory area on which the CRC calculation is performed.

**Bits 1:0  Reserved, must be kept at reset value.**
### 4.9.21 FLASH CRC end address register for bank 1
**(FLASH_CRCEADD1R)**

Address offset: 0x058  
Reset value: 0x0000 0000

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

Bits 31:20  Reserved, must be kept at reset value.  

Bits 19:2  **CRC_END_ADDR[19:2]:** CRC end address on bank 1  

CRC_END_ADDR is used when CRC_BY_SECT is cleared to 0. It must be programmed to the end address of the bank 1 memory area on which the CRC calculation is performed.

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

### 4.9.22 FLASH CRC data register (FLASH_CRCDATAR)

Address offset: 0x05C or 0x15C  
Reset value: 0x0000 0000

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

**CRC_DATA**: CRC result  

CRC_DATA bits contain the result of the last CRC calculation.
4.9.23  **FLASH ECC fail address for bank 1 (FLASH_ECC_FA1R)**

Address offset: 0x060  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>OTP_FAIL_ECC: OTP ECC error bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set to 1 when one single ECC correction or double ECC detection occurred during the last successful read operation from the read-only/ OTP area. The address of the ECC error is available in FAIL_ECC_ADDR1 bitfield.</td>
</tr>
</tbody>
</table>

| Bits 30:16 | Reserved, must be kept at reset value. |
|           |                                   |

| Bits 15:0 | FAIL_ECC_ADDR1: Bank 1 ECC error address |
|          | When an ECC error occurs (both for single correction or double detection) during a read operation from bank 1, the FAIL_ECC_ADDR1 bitfield contains the address that generated the error. |
|          | FAIL_ECC_ADDR1 is reset when the flag error in the FLASH_SR1 register (CLR_SNECCERR1 or CLR_DBECCERR1) is reset. |
|          | The embedded Flash memory programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an ECC error is saved. |
|          | The address in FAIL_ECC_ADDR1 is relative to the Flash memory area where the error occurred (user Flash memory, system Flash memory, read-only/OTP area). |

4.9.24  **FLASH OTP block lock (FLASH_OTPBL_CUR)**

Address offset: 0x068  
Reset value: 0xXXXX XXXX

This register reflects the current values of corresponding option bits.
Embedded Flash memory (FLASH)

4.9.25  FLASH OTP block lock (FLASH_OTPBL_PRG)

Address offset: 0x06C
Reset value: 0xXXXX XXXX

This register is used to program values in corresponding option bits.

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

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **LOCKBL**: OTP Block Lock

Block \( n \) corresponds to OTP 16-bit word 32 x \( n \) to 32 x \( n + 31 \).

LOCKBL[\( n \)] = 1 indicates that all OTP 16-bit words in OTP Block \( n \) are locked and can no longer be programmed.

LOCKBL[\( n \)] = 0 indicates that all OTP 16-bit words in OTP Block \( n \) are not locked and can still be modified.

<table>
<thead>
<tr>
<th>LOCKBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs</td>
</tr>
<tr>
<td>rs</td>
</tr>
<tr>
<td>rs</td>
</tr>
<tr>
<td>rs</td>
</tr>
<tr>
<td>rs</td>
</tr>
<tr>
<td>rs</td>
</tr>
<tr>
<td>rs</td>
</tr>
<tr>
<td>rs</td>
</tr>
<tr>
<td>rs</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **LOCKBL**: OTP Block Lock

Block \( n \) corresponds to OTP 16-bit word 32 x \( n \) to 32 x \( n + 31 \).

LOCKBL[\( n \)] = 1 indicates that all OTP 16-bit words in OTP Block \( n \) are locked and can no longer be programmed.

LOCKBL[\( n \)] = 0 indicates that all OTP 16-bit words in OTP Block \( n \) are not locked and can still be modified.

LOCKBL bits can be set if the corresponding bit in FLASH_OTPBL_CUR is cleared.

4.9.26  FLASH key register for bank 2 (FLASH_KEYR2)

Address offset: 0x104
Reset value: 0x0000 0000

FLASH_KEYR2 is a write-only register. The following values must be programmed consecutively to unlock FLASH_CR2 register and allow programming/erasing it:

1. 1st key = 0x4567 0123
2. 2nd key = 0xCDEF 89AB

This register is available only on STM32H7A3/B3 devices. It is reserved on STM32H7B0.
### Bits 31:0 KEY2R: Bank 2 access configuration unlock key

#### 4.9.27 FLASH control register for bank 2 (FLASH_CR2)

Address offset: 0x10C

Reset value: 0x0000 0001

This register is available only on STM32H7A3/B3 devices. It is reserved on STM32H7B0.

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

**Bits 31:29 Reserved, must be kept at reset value.**

**Bit 28 CRCRDERRIE2:** Bank 2 CRC read error interrupt enable bit

When CRCRDERRIE2 bit is set to 1, an interrupt is generated when a protected area (PCROP or secure-only) has been detected during the last CRC computation on bank 2. CRCRDERRIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated when a CRC read error occurs on bank 2

1: interrupt generated when a CRC read error occurs on bank 2

**Bit 27 CRCENDIE2:** Bank 2 CRC end of calculation interrupt enable bit

When CRCENDIE2 bit is set to 1, an interrupt is generated when the CRC computation has completed on bank 2. CRCENDIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated when CRC computation complete on bank 2

1: interrupt generated when CRC computation complete on bank 2

**Bit 26 DBECCERRIE2:** Bank 2 ECC double detection error interrupt enable bit

When DBECCERRIE2 bit is set to 1, an interrupt is generated when an ECC double detection error occurs during a read operation from bank 2. DBECCERRIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated when an ECC double detection error occurs on bank 2

1: interrupt generated if an ECC double detection error occurs on bank 2
Bit 25 **SNECCERRIE2**: Bank 2 ECC single correction error interrupt enable bit
When SNECCERRIE2 bit is set to 1, an interrupt is generated when an ECC single correction error occurs during a read operation from bank 2. SNECCERRIE2 can be programmed only when LOCK2 is cleared to 0.
0: no interrupt generated when an ECC single correction error occurs on bank 2
1: interrupt generated when an ECC single correction error occurs on bank 2

Bit 24 **RDSERRIE2**: Bank 2 secure error interrupt enable bit
When RDSERRIE2 bit is set to 1, an interrupt is generated when a secure error (access to a secure-only protected address) occurs during a read operation from bank 2. RDSERRIE2 can be programmed only when LOCK2 is cleared to 0.
0: no interrupt generated when a secure error occurs on bank 2
1: an interrupt is generated when a secure error occurs on bank 2

Bit 23 **RDPERRIE2**: Bank 2 read protection error interrupt enable bit
When RDPERRIE2 bit is set to 1, an interrupt is generated when a read protection error occurs (access to an address protected by PCROP or by RDP level 1) during a read operation from bank 2. RDPERRIE2 can be programmed only when LOCK2 is cleared to 0.
0: no interrupt generated when a read protection error occurs on bank 2
1: an interrupt is generated when a read protection error occurs on bank 2

Bit 22 Reserved, must be kept at reset value.

Bit 21 **INCERRIE2**: Bank 2 inconsistency error interrupt enable bit
When INCERRIE2 bit is set to 1, an interrupt is generated when an inconsistency error occurs during a write operation to bank 2. INCERRIE2 can be programmed only when LOCK2 is cleared to 0.
0: no interrupt generated when a inconsistency error occurs on bank 2
1: interrupt generated when a inconsistency error occurs on bank 2.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **STRBERRIE2**: Bank 2 strobe error interrupt enable bit
When STRBERRIE2 bit is set to 1, an interrupt is generated when a strobe error occurs (the master programs several times the same byte in the write buffer) during a write operation to bank 2. STRBERRIE2 can be programmed only when LOCK2 is cleared to 0.
0: no interrupt generated when a strobe error occurs on bank 2
1: interrupt generated when a strobe error occurs on bank 2.

Bit 18 **PGSERRIE2**: Bank 2 programming sequence error interrupt enable bit
When PGSERRIE2 bit is set to 1, an interrupt is generated when a sequence error occurs during a program operation to bank 2. PGSERRIE2 can be programmed only when LOCK2 is cleared to 0.
0: no interrupt generated when a sequence error occurs on bank 2
1: interrupt generated when a sequence error occurs on bank 2.

Bit 17 **WRPERRIE2**: Bank 2 write protection error interrupt enable bit
When WRPERRIE2 bit is set to 1, an interrupt is generated when a protection error occurs during a program operation to bank 2. WRPERRIE2 can be programmed only when LOCK2 is cleared to 0.
0: no interrupt generated when a protection error occurs on bank 2
1: interrupt generated when a protection error occurs on bank 2.
Bit 16 **EOPIE2**: Bank 2 end-of-program interrupt control bit
Setting EOPIE2 bit to 1 enables the generation of an interrupt at the end of a program operation to bank 2. EOPIE2 can be programmed only when LOCK2 is cleared to 0.
0: no interrupt generated at the end of a program operation to bank 2.
1: interrupt enabled when at the end of a program operation to bank 2.

Bit 15 **CRC_EN**: Bank 2 CRC control bit
Setting CRC_EN bit to 1 enables the CRC calculation on bank 2. CRC_EN does not start CRC calculation but enables CRC configuration through FLASH_CRCCR2 register.
When CRC calculation is performed on bank 2, it can only be disabled by setting CRC_EN bit to 0. Resetting CRC_EN clears CRC configuration and resets the content of FLASH_CRC_DATAR register.
CRC_EN can be programmed only when LOCK2 is cleared to 0.

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:6 **SSN2**: Bank 2 sector erase selection number
These bits are used to select the target sector for an erase operation (they are unused otherwise). SSN2 can be programmed only when LOCK2 is cleared to 0.
0x0: Sector 0 of user Flash bank 2 selected
0x1: Sector 1 of user Flash bank 2 selected
...
0x3F: Sector 63 of user Flash bank 2 selected
0x7F: Sector 127 of user Flash bank 2 selected
*Note: Bank 2 is limited to 64 sectors on STM32H7A3xG devices.*

Bit 5 **START2**: Bank 2 erase start control bit
START2 bit is used to start a sector erase or a bank erase operation. START2 can be programmed only when LOCK2 is cleared to 0.
The embedded Flash memory resets START2 when the corresponding operation has been acknowledged. The user application cannot access any embedded Flash memory register until the operation is acknowledged.

Bit 4 **FW2**: Bank 2 write forcing control bit
FW2 forces a write operation even if the write buffer is not full. FW2 can be programmed only when LOCK2 is cleared to 0.
The embedded Flash memory resets FW2 when the corresponding operation has been acknowledged.
Write forcing is effective only if the write buffer is not empty. In particular, FW2 does not start several write operations when the write operations are performed consecutively.

Bit 3 **BER2**: Bank 2 erase request
Setting BER2 bit to 1 requests a bank erase operation on bank 2 (user Flash memory only). BER2 can be programmed only when LOCK2 is cleared to 0.
BER2 has a higher priority than SER2: if both are set, the embedded Flash memory executes a bank erase.
0: bank erase not requested on bank 2
1: bank erase requested on bank 2
*Note: Write protection error is triggered when a bank erase is required and some sectors are protected.*
Bit 2 **SER2**: Bank 2 sector erase request

Setting SER2 bit to 1 requests a sector erase on bank 2. SER2 can be programmed only when LOCK2 is cleared to 0.

BER2 has a higher priority than SER2: if both are set, the embedded Flash memory executes a bank erase.
0: sector erase not requested on bank 2
1: sector erase requested on bank 2

*Note*: Write protection error is triggered when a sector erase is required on protected sector(s).

Bit 1 **PG2**: Bank 2 internal buffer control bit

Setting PG2 bit to 1 enables internal buffer for write operations to bank 2. This allows the preparation of program operations even if a sector or bank erase is ongoing.

PG2 can be programmed only when LOCK2 is cleared to 0. When PG2 is reset, the internal buffer is disabled for write operations to bank 2 and all the data stored in the buffer but not sent to the operation queue are lost.

Bit 0 **LOCK2**: Bank 2 configuration lock bit

This bit locks the FLASH_CR2 register. The correct write sequence to FLASH_KEYR2 register unlocks this bit. If a wrong sequence is executed, or the unlock sequence to FLASH_KEYR2 is performed twice, this bit remains locked until next system reset.

LOCK2 can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK2 changes from 0 to 1, the other bits of FLASH_CR2 register do not change.
0: FLASH_CR2 register unlocked
1: FLASH_CR2 register locked
4.9.28  FLASH status register for bank 2 (FLASH_SR2)

Address offset: 0x110
Reset value: 0x0000 0000

This register is available only on STM32H7A3/B3 devices. It is reserved on STM32H7B0.

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

Bits 31:29  Reserved, must be kept at reset value.

Bit 28  **CRCRDERR2**: Bank 2 CRC read error flag

CRCRDERR2 flag is raised when a word is found read protected during a CRC operation on bank 2. An interrupt is generated if CRCRDIE2 and CRCEND2 are set to 1. Writing 1 to CLR_CRCRDERR2 bit in FLASH_CCR2 register clears CRCRDERR2.

0: no protected area inside the address read by CRC on bank 2
1: a protected area inside the address read by CRC on bank 2. CRC result is very likely incorrect.

**Note:** This flag is valid only when CRCEND2 bit is set to 1.

Bit 27  **CRCEND2**: Bank 2 CRC end of calculation flag

CRCEND2 bit is raised when the CRC computation has completed on bank 2. An interrupt is generated if CRCENDIE2 is set to 1. It is not necessary to reset CRCEND2 before restarting CRC computation. Writing 1 to CLR_CRCEND2 bit in FLASH_CCR2 register clears CRCEND2.

0: CRC computation not complete on bank 2
1: CRC computation complete on bank 2

Bit 26  **DBECCERR2**: Bank 2 ECC double detection error flag

DBECCERR2 flag is raised when an ECC double detection error occurs during a read operation from bank 2. An interrupt is generated if DBECCERRIE2 is set to 1. Writing 1 to CLR_DBECCERR2 bit in FLASH_CCR2 register clears DBECCERR2.

0: no ECC double detection error occurs on bank 2
1: ECC double detection error occurs on bank 2

Bit 25  **SNECCERR2**: Bank 2 single correction error flag

SNECCERR2 flag is raised when an ECC single correction error occurs during a read operation from bank 2. An interrupt is generated if SNECCERRIE2 is set to 1. Writing 1 to CLR_SNECCERR2 bit in FLASH_CCR2 register clears SNECCERR2.

0: no ECC single correction error occurs on bank 2
1: ECC single correction error occurs on bank 2
Bit 24 **RDSERR2**: Bank 2 secure error flag

RDSERR2 flag is raised when a read secure error (read access to a secure-only protected word) occurs on bank 2. An interrupt is generated if RDSERRIE2 is set to 1. Writing 1 to CLR_RDSERR2 bit in FLASH_CCR2 register clears RDSERR2.

0: no secure error occurs on bank 2
1: a secure error occurs on bank 2

Bit 23 **RDPERR2**: Bank 2 read protection error flag

RDPERR2 flag is raised when a read protection error (read access to a PCROP-protected word or a RDP-protected area) occurs on bank 2. An interrupt is generated if RDPERRIE2 is set to 1. Writing 1 to CLR_RDPERR2 bit in FLASH_CCR2 register clears RDPERR2.

0: no read protection error occurs on bank 2
1: a read protection error occurs on bank 2

Bit 22 Reserved, must be kept at reset value.

Bit 21 **INCERR2**: Bank 2 inconsistency error flag

INCERR2 flag is raised when an inconsistency error occurs on bank 2. An interrupt is generated if INCERRIE2 is set to 1. Writing 1 to CLR_INCERR2 bit in the FLASH_CCR2 register clears INCERR2.

0: no inconsistency error occurred on bank 2
1: an inconsistency error occurred on bank 2

Bit 20 Reserved, must be kept at reset value.

Bit 19 **STRBERR2**: Bank 2 strobe error flag

STRBERR2 flag is raised when a strobe error occurs on bank 2 (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if the STRBERRIE2 bit is set to 1. Writing 1 to CLR_STRBERR2 bit in FLASH_CCR2 register clears STRBERR2.

0: no strobe error occurred on bank 2
1: a strobe error occurred on bank 2

Bit 18 **PGSERR2**: Bank 2 programming sequence error flag

PGSERR2 flag is raised when a sequence error occurs on bank 2. An interrupt is generated if the PGSERRIE2 bit is set to 1. Writing 1 to CLR_PGSERR2 bit in FLASH_CCR2 register clears PGSERR2.

0: no sequence error occurred on bank 2
1: a sequence error occurred on bank 2

Bit 17 **WRPERR2**: Bank 2 write protection error flag

WRPERR2 flag is raised when a protection error occurs during a program operation to bank 2. An interrupt is also generated if the WRPERRIE2 is set to 1. Writing 1 to CLR_WRPERR2 bit in FLASH_CCR2 register clears WRPERR2.

0: no write protection error occurred on bank 2
1: a write protection error occurred on bank 2

Bit 16 **EOP2**: Bank 2 end-of-program flag

EOP2 flag is set when a programming operation to bank 2 completes. An interrupt is generated if the EOPIE2 is set to 1. It is not necessary to reset EOP2 before starting a new operation. EOP2 bit is cleared by writing 1 to CLR_EOP2 bit in FLASH_CCR2 register.

0: no programming operation completed on bank 2
1: a programming operation completed on bank 2

Bits 15:4 Reserved, must be kept at reset value.
Bit 3 **CRC_BUSY2**: Bank 2 CRC busy flag

CRC_BUSY2 flag is set when a CRC calculation is ongoing on bank 2. This bit cannot be forced to 0. The user must wait until the CRC calculation has completed or disable CRC computation on bank 2.

0: no CRC calculation ongoing on bank 2
1: CRC calculation ongoing on bank 2.

Bit 2 **QW2**: Bank 2 wait queue flag

QW2 flag is set when a write or erase operation is pending in the command queue buffer of bank 2. It is not possible to know what type of operation is present in the queue. This flag is reset by hardware when all write/erase operations have been executed and thus removed from the waiting queue(s). This bit cannot be forced to 0. It is reset after a deterministic time if no other operations are requested.

0: no write or erase operation is waiting in the operation queues of bank 2
1: at least one write or erase operation is pending in the operation queues of bank 2

Bit 1 **WBNE2**: Bank 2 write buffer not empty flag

WBNE2 flag is set when embedded Flash memory is waiting for new data to complete the write buffer. In this state the write buffer is not empty. WBNE2 is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the event below:

- the application software forces the write operation using FW2 bit in FLASH_CR2
- the embedded Flash memory detects an error that involves data loss
- the application software has disabled write operations in this bank

This bit cannot be forced to 0. To reset it, clear the write buffer by performing any of the above listed actions or send the missing data.

0: write buffer of bank 2 empty or full
1: write buffer of bank 2 waiting data to complete

Bit 0 **BSY2**: Bank 2 busy flag

BSY2 flag is set when an effective write or erase operation is ongoing to bank 2. It is not possible to know what type of operation is being executed.

BSY2 cannot be forced to 0. It is automatically reset by hardware every time a step in a write, or erase operation completes.

0: no write or erase operation is executed on bank 2
1: a write or an erase operation is being executed on bank 2.
### 4.9.29 FLASH clear control register for bank 2 (FLASH_CCR2)

Address offset: 0x114  
Reset value: 0x0000 0000  
This register is available only on STM32H7A3/B3 devices. It is reserved on STM32H7B0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>Res.</td>
<td>CLR_CRCRDERR2</td>
<td>CLR_CRCEND2</td>
<td>CLR_DBECCERR2</td>
<td>CLR_SNECCERR2</td>
<td>CLR_RDSERR2</td>
<td>CLR_RDPERR2</td>
<td>CLR_INCERR2</td>
<td>CLR_STRBERR2</td>
<td>CLR_PGSERR2</td>
<td>CLR_WRPERR2</td>
<td>CLR_EOP2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

- **Bit 28 CLR_CRCRDERR2**: Bank 2 CRCRDERR2 flag clear bit  
  Setting this bit to 1 resets to 0 CRCRDERR2 flag in FLASH_SR2 register.

- **Bit 27 CLR_CRCEND2**: Bank 2 CRCEND2 flag clear bit  
  Setting this bit to 1 resets to 0 CRCEND2 flag in FLASH_SR2 register.

- **Bit 26 CLR_DBECCERR2**: Bank 2 DBECCERR2 flag clear bit  
  Setting this bit to 1 resets to 0 DBECCERR2 flag in FLASH_SR2 register. If the SNECCERR2 flag of FLASH_SR2 register is cleared to 0, FLASH_ECC_FA2R register is reset to 0 as well.

- **Bit 25 CLR_SNECCERR2**: Bank 2 SNECCERR2 flag clear bit  
  Setting this bit to 1 resets to 0 SNECCERR2 flag in FLASH_SR2 register. If the DBECCERR2 flag of FLASH_SR2 register is cleared to 0, FLASH_ECC_FA2R register is reset to 0 as well.

- **Bit 24 CLR_RDSERR2**: Bank 2 RDSERR2 flag clear bit  
  Setting this bit to 1 resets to 0 RDSERR2 flag in FLASH_SR2 register.

- **Bit 23 CLR_RDPERR2**: Bank 2 RDPERR2 flag clear bit  
  Setting this bit to 1 resets to 0 RDPERR2 flag in FLASH_SR2 register.

- **Bit 22 Reserved, must be kept at reset value.**

- **Bit 21 CLR_INCERR2**: Bank 2 INCERR2 flag clear bit  
  Setting this bit to 1 resets to 0 INCERR2 flag in FLASH_SR2 register.

- **Bit 20 Reserved, must be kept at reset value.**

- **Bit 19 CLR_STRBERR2**: Bank 2 STRBERR2 flag clear bit  
  Setting this bit to 1 resets to 0 STRBERR2 flag in FLASH_SR2 register.

- **Bit 18 CLR_PGSERR2**: Bank 2 PGSERR2 flag clear bit  
  Setting this bit to 1 resets to 0 PGSERR2 flag in FLASH_SR2 register.
Bit 17 **CLR_WRPERR2**: Bank 2 WRPERR2 flag clear bit
Setting this bit to 1 resets to 0 WRPERR2 flag in FLASH_SR2 register.

Bit 16 **CLR_EOP2**: Bank 2 EOP2 flag clear bit
Setting this bit to 1 resets to 0 EOP2 flag in FLASH_SR2 register.

Bits 15:0 Reserved, must be kept at reset value.

### 4.9.30 FLASH protection address for bank 2 (FLASH_PRAR_CUR2)

Address offset: 0x128

Reset value: 0xXXXX 0XXX (see Table 20: Option byte organization)

This read-only register reflects the current values of corresponding option bits. It is available only on STM32H7A3/B3 devices and reserved on STM32H7B0.

<table>
<thead>
<tr>
<th>Bit 31-16</th>
<th>PROT_AREA_END2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMEP2</td>
<td></td>
</tr>
<tr>
<td>PROT_AREA_END2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15-0</th>
<th>PROT_AREA_START2</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROT_AREA_START2</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **DMEP2**: Bank 2 PCROP protected erase enable option status bit
If DMEP2 is set to 1, the PCROP protected area in bank 2 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **PROT_AREA_END2**: Bank 2 PCROP area end status bits
These bits contain the last 256-byte block of the PCROP area in bank 2.
If this address is equal to PROT_AREA_START2, the whole bank 2 is PCROP protected.
If this address is lower than PROT_AREA_START2, no protection is set on bank 2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **PROT_AREA_START2**: Bank 2 PCROP area start status bits
These bits contain the first 256-byte block of the PCROP area in bank 2.
If this address is equal to PROT_AREA_END2, the whole bank 2 is PCROP protected.
If this address is higher than PROT_AREA_END2, no protection is set on bank 2.

### 4.9.31 FLASH protection address for bank 2 (FLASH_PRAR_PRG2)

Address offset: 0x12C

Reset value: 0xXXXX 0XXX (see Table 20: Option byte organization)

This register is used to program values in corresponding option bits. It is available only on STM32H7A3/B3 devices and reserved on STM32H7B0.
### 4.9.32 FLASH secure address for bank 2 (FLASH_SCAR_CUR2)

Address offset: 0x130  
Reset value: 0xXXXX 0XXX (see Table 20: Option byte organization)

This read-only register reflects the current values of corresponding option bits. It is available only on STM32H7A3/B3 devices and reserved on STM32H7B0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMEP2</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

#### Bit 31 DMEP2: Bank 2 PCROP protected erase enable option configuration bit
If DMEP2 is set to 1, the PCROP protected area in bank 2 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

#### Bits 30:28 Reserved, must be kept at reset value.

#### Bits 27:16 PROT_AREA_END2: Bank 2 PCROP area end configuration bits
These bits contain the last 256-byte block of the PCROP area in bank 2.  
If this address is equal to PROT_AREA_START2, the whole bank 2 is PCROP protected.  
If this address is lower than PROT_AREA_START2, no protection is set on bank 2.

#### Bits 15:12 Reserved, must be kept at reset value.

#### Bits 11:0 PROT_AREA_START2: Bank 2 PCROP area start configuration bits
These bits contain the first 256-byte block of the PCROP area in bank 2.  
If this address is equal to PROT_AREA_END2, the whole bank 2 is PCROP protected.  
If this address is higher than PROT_AREA_END2, no protection is set on bank 2.
Bit 31 **DMES2**: Bank 2 secure protected erase enable option status bit
   If DMES2 is set to 1, the secure protected area in bank 2 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **SECAREAEND2**: Bank 2 secure-only area end status bits
   These bits contain the last 256-byte block of the secure-only area in bank 2.
   If this address is equal to SECAREASTART2, the whole bank 2 is secure protected.
   If this address is lower than SECAREASTART2, no protection is set on bank 2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **SECAREASTART2**: Bank 2 secure-only area start status bits
   These bits contain the first 256-byte block of the secure-only area in bank 2.
   If this address is equal to SECAREAEND2, the whole bank 2 is secure protected.
   If this address is higher than SECAREAEND2, no protection is set on bank 2.

### 4.9.33 FLASH secure address for bank 2 (FLASH_SCAR_PRG2)

**Address offset**: 0x134

**Reset value**: 0xXXXX 0XXX (see Table 20: Option byte organization)

This register is used to program values in corresponding option bits. It is available only on STM32H7A3/B3 devices and reserved on STM32H7B0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMES2</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>SECAREAEND2</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31 **DMES2**: Bank 2 secure access protected erase enable option configuration bit
   If DMES2 is set to 1, the secure access only area in bank 2 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **SECAREAEND2**: Bank 2 secure-only area end configuration bits
   These bits contain the last of 256 bytes block of the secure-only area in bank 2.
   If this address is equal to SECAREASTART2, the whole bank 2 is secure access only.
   If this address is lower than SECAREASTART2, no protection is set on bank 2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **SECAREASTART2**: Bank 2 secure-only area start configuration bits
   These bits contain the first of 256 bytes block of the secure-only area in bank 2.
   If this address is equal to SECAREAEND2, the whole bank 2 is secure access only.
   If this address is higher than SECAREAEND2, no protection is set on bank 2.
4.9.34  FLASH write sector group protection for bank 2 (FLASH_WPSGN_CUR2R)

Address offset: 0x138
Reset value: 0xXXXX XXXX

This read-only register reflects the current values of corresponding option bits. It is available only on STM32H7A3/B3 devices and reserved on STM32H7B0.

```
<table>
<thead>
<tr>
<th>Bit 31:0</th>
<th>WRPSGn2: Bank 2 sector group protection option status byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>Bit 31:0: Group embedding sectors 0 to 3</td>
</tr>
<tr>
<td></td>
<td>Bit 1: Group embedding sectors 4 to 7</td>
</tr>
<tr>
<td></td>
<td>Bit N: Group embedding sectors 4 ( \times N ) to 4 ( \times N + 3 )</td>
</tr>
<tr>
<td></td>
<td>Bit 31: Group embedding sectors 124 to 127</td>
</tr>
</tbody>
</table>

Note: Bank 2 is limited to 64 sectors on STM32H7A3xG devices.
```

4.9.35  FLASH write sector group protection for bank 2 (FLASH_WPSGN_PRG2R)

Address offset: 0x13C
Reset value: 0xXXXX XXXX

This register is used to program values in corresponding option bits. It is available only on STM32H7A3/B3 devices and reserved on STM32H7B0.

```
<table>
<thead>
<tr>
<th>Bit 31:0</th>
<th>WRPSGn2: Bank 2 sector group protection option status byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>Bit 31:0: Group embedding sectors 0 to 3</td>
</tr>
<tr>
<td></td>
<td>Bit 1: Group embedding sectors 4 to 7</td>
</tr>
<tr>
<td></td>
<td>Bit N: Group embedding sectors 4 ( \times N ) to 4 ( \times N + 3 )</td>
</tr>
<tr>
<td></td>
<td>Bit 31: Group embedding sectors 124 to 127</td>
</tr>
</tbody>
</table>
```
Bits 31:0 **WRPSGn2**: Bank 2 sector group protection option status byte
Setting WRPSGn2 bits to 0 write protects the corresponding group of 4 consecutive sectors in bank 2 (0: group is write protected; 1: group is not write protected)
- Bit 0: Group embedding sectors 0 to 3
- Bit 1: Group embedding sectors 4 to 7
- Bit N: Group embedding sectors 4 x N to 4 x N + 3
- Bit 31: Group embedding sectors 124 to 127

*Note: Bank 2 is limited to 64 sectors on STM32H7A3xG devices.*

### 4.9.36 FLASH CRC control register for bank 2 (FLASH_CRCCCR2)

Address offset: 0x150
Reset value: 0x001C 0000

The values in this register can be changed only if CRC_EN bit is set to 1 in FLASH_CR2 register.

This register is available only on STM32H7A3/B3 devices. It is reserved on STM32H7B0.

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

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

**Bit 22 ALL_BANK**: Bank 2 CRC select bit
When ALL_BANK is set to 1, all bank 2 user sectors are added to the list of sectors on which the CRC is calculated.

**Bits 21:20 CRC_BURST**: Bank 2 CRC burst size
CRC_BURST bits set the size of the bursts that are generated by the CRC calculation unit.
- 00: every burst has a size of 4 Flash words (128 bit)
- 01: every burst has a size of 16 Flash words (128-bit)
- 10: every burst has a size of 64 Flash words (128-bit)
- 11: every burst has a size of 256 Flash words (128-bit)

**Bits 19:18 Reserved, must be kept at reset value.**

**Bit 17 CLEAN_CRC**: Bank 2 CRC clear bit
Setting CLEAN_CRC to 1 clears the current CRC result stored in the FLASH_CRCDBAR register.
Bit 16 **START_CRC**: Bank 2 CRC start bit
START_CRC bit triggers a CRC calculation on bank 2 using the current configuration. It is not possible to start a CRC calculation when an option byte change operation is ongoing because all write accesses to embedded Flash memory registers are put on hold until the option byte change operation has completed.

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

Bit 10 **CLEAN_SECT**: Bank 2 CRC sector list clear bit
Setting CLEAN_SECT to 1 clears the list of sectors on which the CRC is calculated.

Bit 9 **ADD_SECT**: Bank 2 CRC sector select bit
Setting ADD_SECT to 1 adds the sector whose number is CRC_SECT to the list of sectors on which the CRC is calculated.

Bit 8 **CRC_BY_SECT**: Bank 2 CRC sector mode select bit
When CRC_BY_SECT is set to 1, the CRC calculation is performed at sector level, on the sectors selected by CRC_SECT.
When CRC_BY_SECT is reset to 0, the CRC calculation is performed on all addresses between CRC_START_ADDR and CRC_END_ADDR.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **CRC_SECT**: Bank 2 CRC sector number
CRC_SECT is used to select one or more user Flash sectors to be added to CRC calculation. The CRC can be computed either between two addresses (using registers FLASH_CRCSADD2R and FLASH_CRCEADD2R) or on a list of sectors using this register. If this latter option is selected, it is possible to add a sector to the list of sectors by programming the sector number in CRC_SECT and then setting ADD_SECT to 1.
The list of sectors can be erased either by setting CLEAN_SECT bit or by disabling the CRC computation. CRC_SECT can be set only when CRC_EN of FLASH_CR register is set to 1.

0b0000000: sector 0 of bank 2 for CRC
0b0000001: sector 1 of bank 2 for CRC
... 
0b0111111: sector 63 of bank 2 for CRC
... 
0b1111111: sector 127 of bank 2 for CRC

*Note: Bank 2 is limited to 64 sectors on STM32H7AxG devices.*

### 4.9.37 Flash CRC start address register for bank 2
(FLASH_CRCSADD2R)

Address offset: 0x154
Reset value: 0x0000 0000

This register is available only on STM32H7A3/B3 devices. It is reserved on STM32H7B0.
4.9.38  **FLASH CRC end address register for bank 2 (FLASH_CRCEADD2R)**

Address offset: 0x158
Reset value: 0x0000 0000

This register is available only on STM32H7A3/B3 devices. It is reserved on STM32H7B0.

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

Bits 31:20  Reserved, must be kept at reset value.

4.9.39  **FLASH ECC fail address for bank 2 (FLASH_ECC_FA2R)**

Address offset: 0x160
Reset value: 0x0000 0000

This register is available only on STM32H7A3/B3 devices. It is reserved on STM32H7B0.

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

Bits 31:20  Reserved, must be kept at reset value.

Bits 19:2  **CRC_END_ADDR[19:2]**: CRC end address on bank 2

CRC_END_ADDR is used when CRC_BY_SECT is cleared to 0. It must be programmed to the end address of the bank 2 memory area on which the CRC calculation is performed.

Bits 1:0  Reserved, must be kept at reset value.
4.9.40  FLASH register map and reset values

Table 27. Register map and reset value table

| Offset | Register name reset | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x000  | FLASH_ACR           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x004  | FLASH_KEYR1         |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x008  | FLASH_OPTKEYR       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0C   | FLASH_CR1           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x10   | FLASH_SR1           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x14   | FLASH_CCR1          |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  FAIL_ECC_ADDR2: Bank 2 ECC error address

When an ECC error occurs (both for single error correction or double detection) during a read operation from bank 2, the FAIL_ECC_ADDR2 bitfield contains the address that generated the error.

FAIL_ECC_ADDR2 is reset when the flag error in the FLASH_SR2 register (CLR_SNECCERR2 or CLR_DBECCEERR2) is reset.

The embedded Flash memory programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an ECC error is saved.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x018</td>
<td>FLASH_OPTCR</td>
<td>0x00000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x01C</td>
<td>FLASH_OPTSR_CUR</td>
<td>0x00000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x020</td>
<td>FLASH_OPTSR_PRG</td>
<td>0x00000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x024</td>
<td>FLASH_OPTCCR</td>
<td>0x00000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x028</td>
<td>FLASH_PRAR_CUR1</td>
<td>0x00000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x02C</td>
<td>FLASH_PRAR_PRG1</td>
<td>0x00000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x030</td>
<td>FLASH_SCAR_CUR1</td>
<td>0x00000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x034</td>
<td>FLASH_SCAR_PRG1</td>
<td>0x00000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x038</td>
<td>FLASH_WPSGN_CUR1R</td>
<td>0x00000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 27. Register map and reset value table (continued)
### Table 27. Register map and reset value table (continued)

| Offset  | Register name | reset | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|---------|---------------|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x03C   | FLASH_WPSGN_  |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         | PRG1R         |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x040   | FLASH_BOOT_   |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         | CURR          |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x044   | FLASH_BOOT_   |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         | PRGR          |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x048   | Reserved      |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x04C   | Reserved      |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x050   | FLASH_CRCCR1  |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         |               |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x054   | FLASH_CRCADD1 |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         | R            |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x058   | FLASH_CRCADD1 |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         | R            |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x05C   | FLASH_CRCDA  |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         | TUR           |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x060   | FLASH_ECC_   |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         | FA1R          |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x068   | FLASH_OTPBL_  |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         | CUR           |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x06C   | FLASH_OTPBL_  |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         | PRG           |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x100   | FLASH_ACR     |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         |               |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x104   | FLASH_KEYR2   |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         | RER           |       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
Table 27. Register map and reset value table (continued)

| Offset | Register name reset | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x108  | FLASH_OPTKEYR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10C  | FLASH_CR2          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x110  | FLASH_SR2          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x114  | FLASH_CCR2         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x118  | FLASH_OPTCR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x11C  | FLASH_OPTSR_CUR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x120  | FLASH_OPTSR_PRG    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
### Table 27. Register map and reset value table (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x124</td>
<td>FLASH_OPTCCR</td>
<td>0x128</td>
<td>FLASH_PRAR_CUR2</td>
<td>0x12C</td>
<td>FLASH_PRAR_PRG2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x130</td>
<td>FLASH_SCAR_CUR2</td>
<td>0x134</td>
<td>FLASH_SCAR_PRG2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x138</td>
<td>FLASH_WPSGN_CUR2R</td>
<td>0x13C</td>
<td>FLASH_WPSGN_PRG2R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x140</td>
<td>FLASH_BOOT_CURR</td>
<td>0x144</td>
<td>FLASH_BOOT_PRGR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x148</td>
<td>Reserved</td>
<td>0x150</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x154</td>
<td>FLASH_CRCCR2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Value</th>
<th>Offset</th>
<th>Value</th>
<th>Offset</th>
<th>Value</th>
<th>Offset</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x124</td>
<td>0x00000000</td>
<td>0x128</td>
<td>0x00000000</td>
<td>0x130</td>
<td>0x00000000</td>
<td>0x138</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x13C</td>
<td>0x00000000</td>
<td>0x140</td>
<td>0x00000000</td>
<td>0x144</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x148</td>
<td>Reserved</td>
<td>0x150</td>
<td>Reserved</td>
<td>0x154</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Notes:**
- Table 27 provides a register map and reset value table for embedded flash memory (FLASH).
- The table includes columns for offset, register name, and reset value.
- The registers include FLASH_OPTCCR, FLASH_PRAR_CUR2, FLASH_PRAR_PRG2, FLASH_SCAR_CUR2, FLASH_SCAR_PRG2, FLASH_WPSGN_CUR2R, FLASH_WPSGN_PRG2R, FLASH_BOOT_CURR, FLASH_BOOT_PRGR, FLASH_CRCCR2, and FLASH_CRESCADD2R.
- The values represent the reset conditions for each register.

**Key Information:**
- The reset values are crucial for initializing the flash memory correctly.
- Understanding these registers allows for proper configuration and operation of the flash memory in various applications.

---

Refer to the STMicroelectronics documentation for further details and specific application requirements.
Refer to Section 2.3 on page 128 for the register boundary addresses.
5 Secure memory management (SMM)

5.1 Introduction

STM32H7A3/7B3 and STM32H7B0 microcontrollers offer a first set of protection mechanisms, which are similar to other STM32 Series:
- Global readout device protection (RDP)
- Write protection (WRP)
- Proprietary code readout protection (PCROP)

A detailed description of these protection mechanisms is given in Section 4: Embedded Flash memory (FLASH).

STM32H7B3 and STM32H7B0 also offer an additional enhanced protection mode, the Secure access mode, that makes possible the development of user-defined secure services (e.g. secure firmware update or secure boot) and guarantees of a safe execution and protection of both code and data. This mechanism is described in details in Section 5.3: Secure access mode, Section 5.4: Root secure services (RSS) and Section 5.5: Secure user software.

The secure memory management unit is contained inside the CD domain.

5.2 Glossary

The following terms will be used in herein:

<table>
<thead>
<tr>
<th>Table 28. List of preferred terms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Term</strong></td>
</tr>
<tr>
<td><strong>Device Security Level</strong></td>
</tr>
<tr>
<td>Standard mode</td>
</tr>
<tr>
<td>Secure access mode</td>
</tr>
<tr>
<td><strong>Memory areas</strong></td>
</tr>
<tr>
<td>System memory</td>
</tr>
<tr>
<td>User Flash memory</td>
</tr>
<tr>
<td>Secure user memory/area(1)</td>
</tr>
</tbody>
</table>
5.3 Secure access mode

Some sensitive functions require safe execution from potential malicious software attacks. Secure firmware update (SFU) software is a good example of code that requires a high level of protection since it handles secret data (such as cryptographic keys) that shall not be retrieved by other processes.

STM32H7B3 and STM32H7B0 microcontrollers feature secure memory areas with restricted access. They allow building secure services that will be executed prior to any user application. These secure areas, together with the software they contain, are only accessible when configuring the device in Secure access mode.

*Figure 15* gives an overview of Flash memory areas and services in Standard and Secure access modes.

### Table 28. List of preferred terms (continued)

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Software services</strong></td>
<td></td>
</tr>
<tr>
<td>Bootloader</td>
<td>STMicroelectronics software executed at reset which allows the download of firmware from regular communication ports.</td>
</tr>
<tr>
<td>Root secure services (RSS)</td>
<td>STMicroelectronics software which allows the access to secure services.</td>
</tr>
<tr>
<td>Secure user software</td>
<td>User software executed once after reset, which can be used to implement secure boot and secure firmware update (SFU). Secure user software is located in secure user memory.</td>
</tr>
</tbody>
</table>

1. Secure user memory/areas are also named secure-hide protected (HDP) memory/areas.
1. The protected areas that can only be accessed in Secure access mode are shown in blue.
2. A single secure user area can be defined for each bank.
3. Only one bank (bank1) is supported on STM32H7B0 devices.

### 5.3.1 Associated features

The Secure access mode can be configured through option bytes. When it is set, it enables access to:

- STMicroelectronics root secure services to set secure user areas (see Section 5.4: Root secure services (RSS))
- Secure user memory which embeds secure user code and data.

For a summary of access rights for each core, refer to Section 5.6: Summary of Flash protection mechanisms.

### 5.3.2 Boot state machine

In Secure access mode, booting is forced in the RSS whatever the boot configuration (boot pins and boot addresses). The RSS can either set a secure user memory area if one has been requested (see Section 5.5.2: Setting secure user memory areas) or jump directly to the existing secure user memory. The code located in secure user memory is executed before the main user application and the bootloader. If no service is required and no secure area is defined, the RSS jumps to the boot address selected by BOOT0 pin value.

*Figure 16* shows the boot state machine.
4. Only the bank 1 secure user area is available on STM32H7B0 devices. Bank swapping is not supported.

5.3.3 Secure access mode configuration

Enabling Secure access mode

There is no restriction on how to activate Secure access mode on the device. It is configured through the SECURITY option bit in FLASH_OPTSR_CUR register (see Section 4.9.8: FLASH option status register (FLASH_OPTSR_CUR)).

The Secure access mode becomes active after a system reset.

Disabling Secure access mode

Disabling Secure access mode is a more sensitive task as it can only be done if no more protected code exists on the device. As a result, to come back to Standard mode, secure user memories and PCROP/execute-only areas shall be removed before clearing the SECURITY option bit in the FLASH_OPTSR_CUR register.

Protected areas can be removed by performing a Flash mass erase (refer to Section 4.3.10: FLASH erase operations for more details on mass erase sequence).
5.4 Root secure services (RSS)

5.4.1 Secure area setting service

STMicroelectronics provides a service to perform the initialization of secure areas. This service can be called only once. It is executed after a system reset in Secure access mode prior to any other software stored in the device.

RSS software cannot be accessed (read, write, execute and debug) when the STM32H7B3 and STM32H7B0 operate in Standard mode. The service can be automatically accessed with ST programming tool, STM32CubeProgrammer, or called through a direct call to the `resetAndInitializeSecureAreas` function defined below.

**resetAndInitializeSecureAreas**

**Prototype**

```c
void resetAndInitializeSecureAreas(RSS_SecureArea_t area)
```

**Arguments**

Secure user areas start and end addresses. One or two secure user areas can be set.

This service sets secure user area boundaries, following the values stored in the option byte registers:

- SEC_AREA_START1 and SEC_AREA_END1 for bank 1
- SEC_AREA_START2 and SEC_AREA_END2 for bank 2

This service can be used only when a secure area is set for the first time.

A system reset is triggered after service completion.

5.4.2 Secure area exiting service

The RSS also provides the `exitSecureArea` service. This service must be called to jump to user application. It allows closing safely the secure user area to guarantee that its content can no more be accessed.

Contrary to the `resetAndInitializeSecureAreas` service, it does not trigger any system reset. `exitSecureArea` function is defined below:

**exitSecureArea**

**Prototype**

```c
void exitSecureArea(unsigned int vectors, unsigned int jtagState)
```

**Arguments**

Address of application vectors where to jump after exit and state of JTAG after exit:

- RSS_ENABLE_JTAG_AT_EXIT: JTAG enabled after exiting the secure area
- RSS_KEEP_JTAG_DISABLED_AT_EXIT: JTAG disabled after exiting the secure area

This service is used to exit from secure user software and jump to user main application.

There is no system reset triggered by this service
5.4.3 OTFDEC encryption service

The RSS includes the `RSS_OTFD_resetAndEncrypt` service to perform in-place encryption of the provided payload in RAM. Refer to AN5281 “How to use OTFDEC for encryption/decryption in trusted environment on STM32 MCUs” for more details.

5.5 Secure user software

A secure user software is a trusted piece of code that is executed after device power-on or after a system reset. It allows building secure applications such as:

- code signature or integrity checking (user secure boot).
- software license checking
- secure firmware update
- secure initialization

5.5.1 Access rules

Only accessible in Secure access mode, the secure user software is stored in the secure memory areas.

Only one user secure area can be configured per bank. If two secure areas are defined, the secure software that is executed is the one closer to current boot address.

After secure user software execution, the code shall jump to the main user application and prevent access to the secure user area. This is done by calling `exitSecureAreas` secure service with the application code address given as parameter.

Once in the application code, any access to the secure user area triggers a Flash error.

5.5.2 Setting secure user memory areas

One secure area of configurable size can be set in each bank. The size of each area can be set from 512 bytes to full bank with a granularity of 256 bytes:

- Secure area in bank 1
  Boundaries are configured through SEC_AREA_START1 and SEC_AREA_END1 option bits in FLASH_SCAR_CUR1 (see Section 4.9.13: FLASH secure address for bank 1 (FLASH_SCAR_CUR1)).

- Secure area in bank 2
  Boundaries are configured through SEC_AREA_START2 and SEC_AREA_END2 option bits in FLASH_SCAR_CUR2 (see Section 4.9.32: FLASH secure address for bank 2 (FLASH_SCAR_CUR2)).

Note: If the secure area start address is equal to the secure area end address, the whole bank is considered as secure protected.

Flash memory bank 2 is not available on STM32H7B0 devices.

The above option bits can only be initialized through `exitAndInitializeSecureAreas` service.

If a secure area already exists, the secure user area code can update its own secure user area size or create a new one in the other bank.
5.6 Summary of Flash protection mechanisms

Figure 17 and Table 29 summarize the access rights of the different Flash memory areas, both in Secure access and Standard modes.

**Figure 17. Core access to Flash memory areas**

![Diagram showing Flash memory access rights]

1. Flash memory bank 2 is not available on STM32H7B0 devices.

**Table 29. Summary of Flash protected areas access rights**

<table>
<thead>
<tr>
<th>Access type</th>
<th>Software area</th>
<th>Security mode</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution</td>
<td>PCROP</td>
<td>Any</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Secure user software</td>
<td>Secure access</td>
<td>✓(1)</td>
</tr>
<tr>
<td></td>
<td>Root secure services</td>
<td>Secure access</td>
<td>✓(1)</td>
</tr>
<tr>
<td>Read access</td>
<td>PCROP</td>
<td>Any</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Secure user software</td>
<td>Secure access</td>
<td>✓(1)</td>
</tr>
<tr>
<td></td>
<td>Root secure services</td>
<td>Secure access</td>
<td>✓(1)</td>
</tr>
<tr>
<td>Debug access</td>
<td>PCROP</td>
<td>Any</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Secure user software</td>
<td>Secure access</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Root secure services</td>
<td>Secure access</td>
<td>No</td>
</tr>
</tbody>
</table>

1. Access rights granted after reset until code completion only.
6 Power control (PWR)

6.1 Introduction

The power control section (PWR) provides an overview of the supply architecture for the different power domains and of the supply configuration controller.

It also describes the features of the power supply supervisors and explains how the $V_{\text{CORE}}$ supply domain is configured depending on the operating modes, the selected performance (clock frequency) and the voltage scaling.

6.2 PWR main features

- Power supplies and supply domains
  - Core domains ($V_{\text{CORE}}$)
  - $V_{\text{DD}}$ domain
  - External VDD_MMC I/O domain
  - Backup domain ($V_{\text{SW}}, V_{\text{BKP}}$)
  - Analog domain ($V_{\text{DDA}}$)
- System supply voltage regulation
  - SMPS step-down converter
  - Voltage regulator (LDO)
- Peripheral supply regulation
  - USB regulator
- Power supply supervision
  - POR/PDR monitor
  - BOR monitor
  - PVD monitor
  - AVD monitor
  - $V_{\text{BAT}}$ thresholds
  - Temperature thresholds (embedded dedicated temperature monitoring cell)
- Power management
  - $V_{\text{BAT}}$ battery charging
  - Operating modes
  - Voltage scaling control
  - Low-power modes
PWR block diagram

![Power Control Block Diagram](image)

PWR pins and internal signals

Table 30 lists the PWR inputs and output signals connected to package pins or balls, while Table 31 shows the internal PWR signals.

### Table 30. PWR input/output signals connected to package pins or balls

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>Supply input</td>
<td>Main I/O and VDD domain supply input</td>
</tr>
<tr>
<td>VDDA</td>
<td>Supply input</td>
<td>External analog power supply for analog peripherals</td>
</tr>
<tr>
<td>VBAT</td>
<td>Supply input/output</td>
<td>Backup battery supply input or battery charging output</td>
</tr>
<tr>
<td>VDDMMC</td>
<td>Supply input</td>
<td>External power supply for independent I/Os</td>
</tr>
<tr>
<td>VDDSMPS</td>
<td>Supply input</td>
<td>Switched mode power supply input</td>
</tr>
<tr>
<td>VDD50USB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDD33USB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PVD_IN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDDA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSSA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREF+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREF-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB regulator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDDLDO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VCAP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDD60USB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDD33USB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PVD_IN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDDA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSSA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREF+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREF-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB regulator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDDLDO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VCAP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDD60USB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDD33USB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PVD_IN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 30** lists the PWR inputs and output signals connected to package pins or balls, while **Table 31** shows the internal PWR signals.
### Table 30. PWR input/output signals connected to package pins or balls (continued)

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLXSMPS</td>
<td>Supply output</td>
<td>Switched mode power supply output</td>
</tr>
<tr>
<td>VFBSMPS</td>
<td>Supply input</td>
<td>Switched mode power supply feedback voltage sense</td>
</tr>
<tr>
<td>VSSSSMPS</td>
<td>Supply input</td>
<td>Switched mode power supply ground</td>
</tr>
<tr>
<td>VDDLDO</td>
<td>Supply input</td>
<td>Voltage regulator supply input</td>
</tr>
<tr>
<td>VCAP</td>
<td>Supply input/output</td>
<td>Digital core domain supply, generated internally or externally provided</td>
</tr>
<tr>
<td>VREF+</td>
<td>Supply input/output</td>
<td>Reference voltage for ADCs and DACs Generated internally or externally provided</td>
</tr>
<tr>
<td>VREF-</td>
<td>Supply input</td>
<td>Reference voltage for ADCs and DACs</td>
</tr>
<tr>
<td>VDD50USB</td>
<td>Supply input</td>
<td>USB regulator supply input</td>
</tr>
<tr>
<td>VDD33USB</td>
<td>Supply input/output</td>
<td>USB regulator supply output or external USB supply input</td>
</tr>
<tr>
<td>VSS</td>
<td>Supply input</td>
<td>Main ground</td>
</tr>
<tr>
<td>PDR_ON</td>
<td>Digital input</td>
<td>Power-down reset enable</td>
</tr>
<tr>
<td>PVD_IN</td>
<td>Analog input</td>
<td>Monitoring of the voltage level applied to this pin</td>
</tr>
</tbody>
</table>

### Table 31. PWR internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHB</td>
<td>Digital I/O</td>
<td>AHB register interface</td>
</tr>
<tr>
<td>pwr_pvd_wkup</td>
<td>Digital output</td>
<td>Programmable voltage detector output</td>
</tr>
<tr>
<td>pwr_avd_wkup</td>
<td>Digital output</td>
<td>Analog voltage detector output</td>
</tr>
<tr>
<td>pwr_por_rst</td>
<td>Digital output</td>
<td>Power-on reset</td>
</tr>
<tr>
<td>pwr_bor_rst</td>
<td>Digital output</td>
<td>Brownout reset</td>
</tr>
<tr>
<td>exti_c_wkup</td>
<td>Digital input</td>
<td>CPU wakeup request</td>
</tr>
<tr>
<td>exti_srd_wkup</td>
<td>Digital input</td>
<td>SmartRun domain wakeup request</td>
</tr>
<tr>
<td>pwr_cd_wkup</td>
<td>Digital output</td>
<td>CPU bus matrix and CPU peripherals clock wakeup request</td>
</tr>
<tr>
<td>pwr_srd_wkup</td>
<td>Digital output</td>
<td>SmartRun domain bus matrix clock wakeup request</td>
</tr>
<tr>
<td>rcc_pwd_srd_req</td>
<td>Digital input</td>
<td>SmartRun domain low-power request generated by the RCC</td>
</tr>
<tr>
<td>rcc_pwd_cd_req</td>
<td>Digital input</td>
<td>CPU domain low-power request generated by the RCC</td>
</tr>
</tbody>
</table>
6.4 Power supplies

The device requires \( V_{\text{DD}}, V_{\text{DDLDO}}, V_{\text{DDA}} \) and \( V_{\text{BAT}} \) power supplies. Depending on the use case and when available from the package, \( V_{\text{DDSMPS}}, V_{\text{DDMMC}}, V_{\text{DDUSB}}, V_{\text{REF+}} \) and \( V_{\text{CAP}} \) independent power supplies can also be required.

The device provides regulated supplies for specific functions (SMPS, voltage regulator, USB regulator, voltage reference buffer):

- \( V_{\text{DD}} \) external power supply for I/Os and system analog blocks such as reset, power management and oscillators
- \( V_{\text{DDMMC}} \) external power supply for some independent I/Os (available from some specific packages only), must be tied to \( V_{\text{DD}} \) when an independent supply is not required
- \( V_{\text{BAT}} \) optional external power supply for Backup domain when \( V_{\text{DD}} \) is not present (\( V_{\text{BAT}} \) mode), must be connected to \( V_{\text{DD}} \) when this feature is not used
- \( V_{\text{DDSMPS}} \) external power supply for the switched mode power supply. This power supply must be connected to \( V_{\text{DD}} \) or tied to \( V_{\text{SS}} \) when the SMPS is not used.
- \( V_{\text{LXSMPS}} \) switched mode power supply output
- \( V_{\text{FBMS}} \) is the switched mode power supply sense feedback, must be tied to \( V_{\text{SS}} \) when the SMPS is not used
- \( V_{\text{SSSMPS}} \) separate switched mode power supply ground
- \( V_{\text{DDLDO}} \) external power supply for the voltage regulator
- \( V_{\text{CAP}} \) digital core domain supply

This power supply is independent from all the other power supplies:

- When the voltage regulator is enabled, \( V_{\text{CORE}} \) is delivered by the internal voltage regulator.
- When the voltage regulator is disabled, \( V_{\text{CORE}} \) is delivered by an external power supply through \( V_{\text{CAP}} \) pin, or by the switched mode power supply.

- \( V_{\text{DDA}} \) external analog power supply for ADCs, DACs, OPAMPs, comparators and voltage reference buffers

This power supply is independent from all the other power supplies.

- \( V_{\text{REF+}} \) external reference voltage for ADC and DAC

  - When the voltage reference buffer is enabled, \( V_{\text{REF+}} \) is delivered by the internal voltage reference buffer.
  - When the voltage reference buffer is disabled, \( V_{\text{REF+}} \) is delivered by an independent external reference supply (**Do not** enable the \( V_{\text{REF}} \) buffer in this case).

- \( V_{\text{SSA}} \) separate analog and reference voltage ground
- \( V_{\text{DD50USB}} \) external power supply for USB regulator
- \( V_{\text{DD33USB}} \) USB regulator supply output for USB interface

  - When the USB regulator is enabled, \( V_{\text{DD33USB}} \) is delivered by the internal USB regulator.
  - When the USB regulator is disabled, \( V_{\text{DD33USB}} \) is delivered by an independent external supply input (**Do not** enable the USB regulator in this case).

- \( V_{\text{SS}} \) common ground for all supplies except for SMPS and analog blocks
Note: Depending on the operating power supply range, some peripherals may be used with limited features and performance. For more details, refer to section "General operating conditions" of the device datasheets.
Figure 19. Power supply overview

- **Core domain**
  - VDD domain
  - Backup domain
  - Analog domain

- **CPU domain**
  - Periphery domain
    - CPU
    - Flash

- **Backup domain**
  - LSI, LSE, RTC, wakeup logic, backup registers, reset
  - Backup RAM

- **Analog domain**
  - OPAMP, comparator

- **Hardware components**
  - VDDSMPS, VLXSMPS, VFBSMPS, VSSSMPS
  - VCAP
  - VDDLDO, VSS
  - VDDMMC
  - VDD
  - VBAT, VB
  - BKUP
  - VDDA
  - VREF+, VREF-, VSSA

- **Power supplies**
  - VCORE, VSS
  - VDDLDO
  - VDDSMPS, VLXSMPS, VFBSMPS, VSSSMPS
  - VDD
  - VBAT
  - VDDA

- **Regulators**
  - Voltage regulator
  - USB regulator
  - Level shifter
  - ADC, DAC
  - Flash
  - Power switch

- **Other components**
  - SmartRun domain
    - (system logic, AIEC, peripherals, RAM)
  - SRD
  - Always ON (retention, RAM arrays)
  - DStop2 (deep retention)
  - HSI, CSI, HSI48, HSE, PLLs
  - LSI, LSE, RTC, wakeup logic, backup registers, reset
  - IOs
  - OPAMP, comparator
  - USB
  - Power switch

- **Power modes**
  - Standby
  - Active
  - Deep sleep

- **Power domains**
  - Analog
  - Core
  - CPU
  - Backup
  - Always ON

- **Power switch**
  - On/Off

- **Voltage levels**
  - VDD, VBAT, VSS

- **Supply types**
  - Switched mode power supply (SMPS)
  - USB
  - Other types
By configuring the switched mode power supply (SMPS) and the voltage regulator, the supply configurations shown in Figure 20 are supported for the $V_{CORE}$ core domain and an external supply.

Note: The SMPS is not available on all packages.

Figure 20. System supply configurations

Note: For cases 3 to 5, the SMPS output is set to 1.2 V during the startup phase and to 1.8 V or 2.5 V at code execution start (refer to PWR control register 3 (PWR_CR3)). The different supply configurations are controlled through the LDOEN, SMPSEN, SMPSEXTHP, SMPSLEVEL and BYPASS bits in the PWR control register 3 (PWR_CR3), according to Table 32.
## Table 32. Supply configuration control

<table>
<thead>
<tr>
<th>ID</th>
<th>Supply configuration</th>
<th>SMPSLEVEL</th>
<th>SMPSEN</th>
<th>SMPSEXTHP</th>
<th>LDOEN</th>
<th>BYPASS</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0  | Startup configuration | 00        | 0      | 1         | 1     | 0      | – Configuration during power-up phase (not a user config)  
– \(V_{\text{CORE}}\) power domains supplied from the LDO (VOS3, 1 V)  
– SMPS enabled at 1.2 V |
| 1  | LDO supply           | \(x\)     | \(x\)  | 0         | 1     | 0      | – \(V_{\text{CORE}}\) power domains are supplied from the LDO according to VOS.  
– LDO power mode (Main, LP, Off) follows system low-power modes.  
– SMPS disabled |
| 2  | Direct SMPS supply   | \(x\)     | 0      | 1         | 0     | 0      | – \(V_{\text{CORE}}\) power domains are supplied from the SMPS according to VOS.  
– LDO bypassed  
– SMPS power mode (MR, LP, Off) follows system low-power modes. |
| 3  | SMPS supplies LDO    | 01 or 10  | 0      | 1         | 1     | 0      | – \(V_{\text{CORE}}\) power domains are supplied from the LDO according to VOS.  
– LDO power mode (Main, LP, Off) follows system low-power modes.  
– SMPS is enabled according to SMPSLEVEL and supplies the LDO.  
– SMPS power mode (MR, LP or Off) follows the system low-power mode. |
| 4  | SMPS supplies external and LDO | 01 or 10 | 1      | 1         | 1     | 0      | – \(V_{\text{CORE}}\) power domains are supplied from voltage regulator according to VOS  
– LDO power mode (Main, LP, Off) follows system low-power modes.  
– SMPS enabled according to SMPSLEVEL, used to supply external circuits and may supply the LDO.  
– SMPS forced ON in MR mode |
| 5  | SMPS supplies external and LDO Bypass | 01 or 10 | 1      | 1         | 0     | 1      | – \(V_{\text{CORE}}\) supplied from external source  
– SMPS enabled according to SMPSLEVEL, used to supply external circuits and may supply the external source for \(V_{\text{CORE}}\).  
– SMPS converter forced ON in MR mode |
| 6  | SMPS disabled and LDO Bypass | \(x\) | \(x\) | 0         | 0     | 1      | – \(V_{\text{CORE}}\) supplied from external source  
– SMPS disabled and LDO bypassed, voltage monitoring still active |
6.4.1 System supply startup

The system startup sequence from power-on in different supply configurations is the following (see Figure 21 for LDO supply and Figure 23 for direct SMPS supply):

1. When the system is powered on, the POR monitors the V_DD supply. Once VDD is above the POR threshold level, the SMPS and the voltage regulator are enabled in the default supply configuration:
   - The SMPS output level is set at 1.2 V.
   - The voltage converter output level is set at 1.0 V in accordance with the three levels configured in PWR SmartRun domain control register (PWR_SRDCR).
2. The system is kept in reset mode as long as V_CORE is not correct.
3. Once V_CORE is correct, the system is taken out of reset and the HSI oscillator is enabled.
4. Once the oscillator is stable, the system is initialized: Flash memory and option bytes are loaded and the CPU starts in limited run mode (Run*).
5. The software must then initialize the system including supply configuration programming in PWR control register 3 (PWR_CR3). Once the supply configuration has been configured, the ACTVOSRDY bit in PWR control status register 1 (PWR_CSR1) must be checked to guarantee valid voltage levels:
   a) As long as ACTVOSRDY indicates that voltage levels are invalid, the system is in Run* mode: write accesses to the RAMs are not permitted and VOS must not be changed.
   b) Once ACTVOSRDY indicates that voltage levels are valid, the system is in normal Run mode. Write accesses to RAMs are allowed and VOS can be changed.

Startup with V_CORE supplied from the LDO voltage regulator

When V_CORE is supplied from the voltage regulator (LDO), the VCORE voltage settles directly at VOS3 level. However the SMPS VFBSMPS output voltage is set at 1.2 V. The ACTVOSRDY bit in PWR control status register 1 (PWR_CSR1) indicates that the voltage levels are invalid.

<table>
<thead>
<tr>
<th>ID</th>
<th>Supply configuration</th>
<th>SMPSLEVEL</th>
<th>SMPSEXTHP</th>
<th>SMPSEN</th>
<th>LDOEN</th>
<th>BYPASS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>Illegal</td>
<td>xx x 0 0 0</td>
<td>x x 1 1 0</td>
<td>0 0 x 1 1</td>
<td>xx 1 1 0 0</td>
<td>0 0 1 1 0</td>
<td>If an illegal combination is written, the default configuration is kept (startup configuration, writing ignored) and a power-on reset (POR) is required before writing a new combination.</td>
</tr>
</tbody>
</table>

Table 32. Supply configuration control (continued)
The software must program the supply configuration in PWR control register 3 (PWR_CR3). In addition, the V_FB8MS voltage level must reach the programmed SMPSLEVEL so that ACTVOSRDY indicates a valid voltage level (see Figure 21).

Figure 21. Device startup with V\textsubscript{CORE} supplied from voltage regulator

1. In Run\textsuperscript{*} mode, write operations to RAM are not allowed.
2. Write operations to RAM are allowed and VOS can be changed only when ACTVOSRDY is valid.

When exiting from Standby mode, the supply configuration is known by the system since the content of the PWR control register 3 (PWR_CR3) is retained. However, the software must still wait for the ACTVOSRDY bit to be set in PWR control status register 1 (PWR_CSR1) to indicate V\textsubscript{CORE} voltage levels are valid, before performing write accesses to RAM or changing VOS.

**Startup with V\textsubscript{CORE} supplied directly from the SMPS**

When V\textsubscript{CORE} is supplied directly from the SMPS, the V\textsubscript{CORE} voltage first settles at the SMPS V\textsubscript{FB8MS} default level (1.2 V). Due to a too high supply compared to the VOS3 level, the ACTVOSRDY bit in PWR control status register 1 (PWR_CSR1) indicates an invalid
voltage levels. \( V_{\text{CORE}} \) settles at 1.0 V (VOS3 level) and ACTVODSRDY indicates a valid voltage levels only when the supply configuration has been programmed in **PWR control register 3 (PWR_CR3)** (see Figure 5).

**Figure 22. Device startup with \( V_{\text{CORE}} \) supplied from SMPS**

1. In Run* mode, write operations to RAM are not allowed.
2. Write operations to RAM are allowed and VOS can be changed only when ACTVOSRDY is valid.

**Startup with \( V_{\text{CORE}} \) provided from an external supply (Bypass)**

Once \( V_{\text{DD}} \) is above the POR threshold level, the voltage regulator is enabled and sets the output level provided to the core domain to 1.0 V.

For this reason, the external supply provided to the core domain needs to be available before the internal voltage converter starts, to insure the voltage converter output stays switched off.

At code execution start, the voltage converter is switched off.
6.4.2 Core domain

The $V_{\text{CORE}}$ core domain supply can be provided by the SMPS, by the voltage regulator or by an external supply (through the VCAP pads). $V_{\text{CORE}}$ supplies all the digital circuitries except for the Backup domain and the Standby circuitry. The $V_{\text{CORE}}$ domain is split into two sections:

- CPU domain (CD) containing the CPU (Cortex®-M7), Flash memory and peripherals
- SmartRun domain (SRD) containing the system control, I/O logic and low-power peripherals

When a power-on reset occurs, the voltage regulator is enabled and supplies $V_{\text{CORE}}$. The SMPS is also enabled to deliver 1.2 V. This allows the system to start up in any supply configurations (see Figure 20).

After a power-on reset, the software must configure the used supply configuration in the PWR control register 3 (PWR_CR3) before changing VOS in the PWR SmartRun domain control register (PWR_SRDCR) or the RCC $ck_{\text{sys}}$ frequency. The different system supply configurations are controlled as shown in Table 32.

Note: The SMPS is not available on all packages.

Voltage regulator

The embedded voltage regulator (LDO) requires external capacitors to be connected to VCAP pins.

The voltage regulator provides three different operating modes: Main (MR), Low-power (LP) or Off. These modes are used depending on the system operating modes (Run, Stop and Standby).

- Run mode
  The LDO regulator is in Main mode and provides full power to the $V_{\text{CORE}}$ domain (core, memories and digital peripherals). The regulator output voltage can be scaled by software to different voltage levels (VOS0, VOS1, VOS2, and VOS3) that are configured through VOS bits in PWR SmartRun domain control register (PWR_SRDCR). The VOS voltage scaling allows optimizing the power consumption when the system is clocked below the maximum frequency. By default VOS3 is selected after system reset. VOSx bits can be changed on-the-fly to adapt to the required system performance.

- Stop mode
  The voltage regulator supplies the $V_{\text{CORE}}$ domain to retain the content of registers and internal memories.
  The regulator can be kept in Main mode to allow fast exit from Stop mode or can be set in LP mode to obtain a lower $V_{\text{CORE}}$ supply level and extend the exit-from-Stop latency. The regulator mode is selected through the SVOS and LPDS bits in PWR control register 1 (PWR_CR1). Main mode and LP mode are allowed if SVOS3 voltage scaling is selected, while only LP mode is possible for SVOS4 and SVOS5 scaling. Due to a lower voltage level for SVOS4 and SVOS5 scaling, the Stop mode consumption can be further reduced.

- Standby mode
  The voltage regulator is OFF and the $V_{\text{CORE}}$ domains are powered down. The content of the registers and memories is lost except for the Standby circuitry and the Backup domain.
**SMPS regulator**

The switched mode power supply (SMPS) requires an external coil to be connected between the dedicated VLXSMPS pin and VSS via a capacitor.

The SMPS can be used in internal supply mode or external supply mode. The internal supply mode is used to directly supply the $V_{CORE}$ domain, while the external supply mode is used to generate an intermediate supply level ($V_{DD\_Extern}$ at 1.8 or 2.5 V) that can supply the voltage regulator and optionally an external circuitry.

The SMPS works in three different power modes: Main (MR), Low-power (LP) or Off.

When the SMPS is used in internal supply mode, the converter operating modes depend on the system modes (Run, Stop, Standby) and are configured through the associated VOS and SVOS levels:

- **Run mode**
  The SMPS operates in MR mode and provides full power to the $V_{CORE}$ domain (core, memories and digital peripherals). The regulator output voltage can be scaled by software to different voltage levels (VOS0, VOS1, VOS2, and VOS3) that are configured through the VOS bits in the *PWR SmartRun domain control register (PWR_SRDCR)*. The VOS voltage scaling allows optimizing the power consumption when the system is clocked below the maximum frequency. By default VOS3 is selected after system reset. VOSx bits can be changed on-the-fly to adapt to the required system performance.

- **Stop mode**
  The SMPS supplies the $V_{CORE}$ domain to retain the content of registers and internal memories. The regulator can be kept in MR mode to allow fast exit from Stop mode, or can be set in LP mode to achieve a lower $V_{CORE}$ supply level but an extend exit-from-Stop latency. The regulator mode is selected through the SVOS and LPDS bits in the *PWR control register 1 (PWR_CR1)*. MR mode or LP mode are allowed if SVOS3 voltage scaling is selected, while only LP mode is possible for SVOS4 and SVOS5 scaling. Due to a lower voltage level for SVOS4 and SVOS5 scaling, the Stop mode power consumption can be further reduced.

- **Standby mode**
  The SMPS is OFF and the $V_{CORE}$ domains are powered down. The content of the registers and memories are lost except for the Standby circuitry and the Backup domain.

When the SMPS supplies an external circuitry by generating an intermediate voltage level, the converter is forced ON and operates in MR mode. The intermediate voltage level is selected through SMPSLEVEL bits in the *PWR control register 3 (PWR_CR3)*. $V_{DD\_Extern}$ is supplied at all times with full power whatever the system modes (Run, Stop, Standby).

**Note:** The SMPS is not available on all packages.
6.4.3 **PWR external supply**

When \( V_{\text{CORE}} \) is supplied from an external source, different operating modes can be used depending on the system operating modes (Run, Stop or Standby):

- **In Run mode**
  
The external source supplies full power to the \( V_{\text{CORE}} \) domain (core, memories and digital peripherals). The external source output voltage is scalable through different voltage levels (VOS0, VOS1, VOS2 and VOS3). The externally applied voltage level must be reflected in the VOSx bits in the **PWR SmartRun domain control register (PWR_SRDCR)**. The RAMs must only be accessed for write operations when the external applied voltage level matches VOS settings.

- **In Stop mode**
  
The external source supplies \( V_{\text{CORE}} \) domain to retain the content of registers and internal memories. The regulator can select a lower \( V_{\text{CORE}} \) supply level to reduce the consumption in Stop mode.

- **In Standby mode**
  
The external source must be switched OFF and the \( V_{\text{CORE}} \) domains powered down. The content of registers and memories is lost except for the Standby circuitry and the Backup domain. The external source must be switched ON when exiting Standby mode.

  Care must be taken that all the current operations and transfers are completed before entering Standby and switching OFF the external source.

6.4.4 **Backup domain**

To retain the content of the Backup domain (RTC, backup registers and backup RAM) when \( V_{\text{DD}} \) is turned off, \( V_{\text{BAT}} \) pin can be connected to an optional voltage that is supplied from a battery or from another source.

The switching to \( V_{\text{BAT}} \) is controlled by the power-down reset embedded in the reset block that monitors the \( V_{\text{DD}} \) supply.

---

**Warning:** During \( t_{\text{RSTTEMPO}} \) (temporization at \( V_{\text{DD}} \) startup) or after a PDR is detected, the power switch between \( V_{\text{BAT}} \) and \( V_{\text{DD}} \) remains connected to \( V_{\text{BAT}} \). During the startup phase, if \( V_{\text{DD}} \) is established in less than \( t_{\text{RSTTEMPO}} \) (see the datasheet for the value of \( t_{\text{RSTTEMPO}} \)) and \( V_{\text{DD}} > V_{\text{BAT}} + 0.6 \text{ V} \), a current may be injected into \( V_{\text{BAT}} \) pin through an internal diode connected between \( V_{\text{DD}} \) and the power switch (\( V_{\text{BAT}} \)). If the power supply/battery connected to the \( V_{\text{BAT}} \) pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the \( V_{\text{BAT}} \) pin.
When the VDD supply is present, the Backup domain is supplied from VDD. This allows saving VBAT power supply battery life time.

If no external battery is used in the application, it is recommended to connect VBAT externally to VDD through a 100 nF external ceramic capacitor.

When the VDD supply is present and higher than the PDR threshold, the Backup domain is supplied by VDD and the following functions are available:

- PC14 and PC15 can be used either as GPIO or as LSE pins.
- PC13 can be used either as GPIO or as RTC_OUT1, RTC_TS, TAMP_IN1, TAMP_OUT2 or TAMP_OUT3 pin assuming they have been configured by the RTC or the TAMPER.
- PI8 can be used either as GPIO or as RTC_OUT2, TAMP_IN2 or TAMP_OUT3 pin assuming they have been configured by the RTC or the TAMPER.
- PC1 can be used as TAMP_IN3 assuming it has been configured by the TAMPER.

Note: Since the switch only sinks a limited amount of current, the use of PC13 to PC15 and PI8 GPIOs is restricted: only one I/O can be used as an output at a time, at a speed limited to 2 MHz with a maximum load of 30 pF. These I/Os must not be used as current sources (e.g. to drive an LED).

In VBAT mode, when the VDD supply is absent and a supply is present on VBAT, the Backup domain is supplied by VBAT and the following functions are available:

- PC14 and PC15 can be used as LSE pins only.
- PC13 can be used as RTC_OUT1, RTC_TS, TAMP_IN1, TAMP_OUT2 or TAMP_OUT3 pin assuming they have been configured by the RTC or the TAMPER.
- PI8 can be used as RTC_OUT2, TAMP_IN2 or TAMP_OUT3 pin assuming they have been configured by the RTC or the TAMPER.

Accessing the Backup domain

After reset, the Backup domain (RTC registers and RTC backup registers) is protected against possible unwanted write accesses. To enable access to the Backup domain, set the DBP bit in the **PWR control register 1 (PWR_CR1)**.

For more detail on RTC and backup RAM access, refer to **Section 8: Reset and clock control (RCC)**.

Backup RAM

The Backup domain includes 4 Kbytes of backup RAM accessible in 32-, 16- or 8-bit data mode. The backup RAM is supplied from the backup regulator in the Backup domain. When the backup regulator is enabled through BREN bit in the **PWR control register 2 (PWR_CR2)**, the backup RAM content is retained even in Standby and/or VBAT mode (it can be considered as an internal EEPROM if VBAT is always present).

The backup regulator can be ON or OFF depending whether the application needs the backup RAM function in Standby or VBAT modes.

The backup RAM is not mass erased by an tamper event, instead it is read protected to prevent confidential data, such as cryptographic private key, from being accessed. To re-
gain access to the backup RAM after a tamper event, the memory area needs to be first erased. The backup RAM can be erased in the two following ways:

- through the Flash interface when a protection level change from level 1 to level 0 is requested (refer to the description of read protection (RDP) in the Flash programming manual).
- after a tamper event, by filling the backup RAM with zeros upon the first attempt to write access it

**Figure 23. Backup domain**

### 6.4.5 V\textsubscript{BAT} battery charging

When V\textsubscript{DD} is present, the external battery connected to V\textsubscript{BAT} can be charged through an internal resistance.

V\textsubscript{BAT} charging can be performed either through a 5 k\textohm resistor or through a 1.5 k\textohm resistor, depending on the VBRS bit value in **PWR control register 3 (PWR\_CR3)**.

The battery charging is enabled by setting the VBE bit in **PWR control register 3 (PWR\_CR3)**. It is automatically disabled in V\textsubscript{BAT} mode.

### 6.4.6 Analog supply

**Separate V\textsubscript{DDA} analog supply**

The analog supply domain is powered by dedicated VDDA and VSSA pads that allow the supply to be filtered and shielded from noise on the PCB, thus improving ADC and DAC conversion accuracy:

- The analog supply voltage input is available on a separate VDDA pin.
- An isolated supply ground connection is provided on VSSA pin.
Analog reference voltage $V_{REF+}/V_{REF-}$.

To achieve better accuracy low-voltage signals, the ADC and DAC have a separate reference voltage, available on VREF+ pin. The user can connect a separate external reference voltage on VREF+ pin.

The $V_{REF+}$ controls the highest voltage, represented by the full scale value, the lower voltage reference ($V_{REF-}$) being connected to VSSA pin.

When enabled by ENVR bit in the VREFBUF control and status register (see Section 30: Voltage reference buffer (VREFBUF)), $V_{REF+}$ is provided from the internal voltage reference buffer. The internal voltage reference buffer can also deliver a reference voltage to external components through VREF+ pin.

When the internal voltage reference buffer is disabled by ENVR, $V_{REF+}$ needs to be delivered by an independent external reference supply voltage or connected with $V_{DDA}$.

Note: The VREF+ and VREF- pins are not available on all packages (connected internally respectively to VDDA and VSSA). Do not enable the internal voltage reference buffer when an external power supply is applied to the VREF+ pin.

6.4.7 USB regulator

The USB transceiver is supplied from a dedicated $V_{DD33USB}$ supply that can be provided either by the integrated USB regulator or by an external USB supply.

When enabled by USBREGEN bit in PWR control register 3 (PWR_CR3), the $V_{DD33USB}$ is provided from the USB regulator. Before using $V_{DD33USB}$, check that it is available by monitoring USB33RDY bit in PWR control register 3 (PWR_CR3). The $V_{DD33USB}$ supply level detector must be enabled through USB33DEN bit in PWR control register 3 (PWR_CR3).

When the USB regulator is disabled through USBREGEN bit, $V_{DD33USB}$ can be provided from an external supply. In this case $V_{DD33USB}$ and $V_{DD50USB}$ must be connected together.

For more information on the USB regulator (see Section 62: USB on-the-go high-speed (OTG_HS)).
6.5 Power supply supervision

Power supply level monitoring is available on the following supplies:

- $V_{DD} (V_{DDSMPS})$ via POR/PDR (see Section 6.5.1), BOR (see Section 6.5.2) and PVD monitor (see Section 6.5.3)
- $V_{DDA}$ via AVD monitor (see Section 6.5.4)
- $V_{BAT}$ via $V_{BAT}$ threshold (see Section 6.5.5)
- $V_{CORE}$ over-voltage protection (see Section 6.5.7)
- $V_{SW}$ via $\text{rst}_{\text{vsw}}$, keeping $V_{SW}$ domain in Reset mode as long as the level is not OK
- $V_{BKP}$ via a BRRDY bit in PWR control register 2 (PWR_CR2)
- $V_{FB_{SMPS}}$ via a SMPSEXTRDY bit in PWR control register 3 (PWR_CR3)
- $V_{DD33USB}$ via USB33RDY bit in PWR control register 3 (PWR_CR3)
- $V_{DDMMC}$ via MMCVDO bit in PWR control register 1 (PWR_CR1)

6.5.1 Power-on reset (POR)/power-down reset (PDR)

The system has an integrated POR/PDR circuitry that ensures proper startup operation.

The system remains in Reset mode when $V_{DD}$ is below a specified $V_{POR}$ threshold, without the need for an external reset circuit. Once the $V_{DD}$ supply level is above the $V_{POR}$ threshold, the system is taken out of reset (see Figure 25). For more details concerning the power-on/power-down reset threshold, refer to the electrical characteristics section of the datasheets.

The POR/PDR can be enabled/disabled by the device PDR_ON input pin.

### Figure 25. Power-on reset/power-down reset waveform

1. For thresholds and hysteresis values, refer to the datasheets.
6.5.2 Brownout reset (BOR)

During power-on, the brownout reset (BOR) keeps the system under reset until the $V_{DD}$ supply voltage reaches the specified $V_{BOR}$ threshold.

The $V_{BOR}$ threshold is configured through system option bytes. By default, BOR is OFF. The following programmable $V_{BOR}$ thresholds can be selected:

- BOR off
- BOR level 1 ($V_{BOR1}$)
- BOR level 2 ($V_{BOR2}$)
- BOR level 3 ($V_{BOR3}$)

For more details on the brownout reset thresholds, refer to the section “Electrical characteristics” of the product datasheets.

A system reset is generated when the BOR is enabled and $V_{DD}$ supply voltage drops below the selected $V_{BOR}$ threshold.

BOR can be disabled by programming the system option bytes. To disable the BOR function, $V_{DD}$ must have been higher than the POR threshold to start the system option byte programming sequence. The power-down is then monitored by the PDR (see Section 6.5.1).

Figure 26. BOR thresholds

![BOR thresholds diagram](MSv40341V2)

1. For thresholds and hysteresis values, refer to the datasheets.

6.5.3 Programmable voltage detector (PVD)

The PVD can be used to monitor the $V_{DD}$ power supply by comparing it to a threshold selected by the PLS[2:0] bits in the PWR control register 1 (PWR_CR1). The PVD can also
be used to monitor a voltage level on the PVD_IN pin. In this case PVD_IN voltage is compared to the internal VREFINT level.

The PVD is enabled by setting the PVDE bit in PWR control register 1 (PWR_CR1).

A PVDO flag is available in the PWR control status register 1 (PWR_CSR1) to indicate if V_DD or PVD_IN voltage is higher or lower than the PVD threshold. This event is internally connected to the EXTI and can generate an interrupt, provided it has been enabled through the EXTI registers. The rising/falling edge sensitivity of the EXTI line must be configured according to PVD output behavior, i.e. if the EXTI line is configured to rising edge sensitivity, the interrupt is generated when V_DD or PVD_IN voltage drops below the PVD threshold. As an example, the service routine could perform emergency shutdown.

**Figure 27. PVD thresholds**

![PVD thresholds diagram](MSv40342V2)

1. For thresholds and hysteresis values, refer to the datasheets.
6.5.4 Analog voltage detector (AVD)

The AVD can be used to monitor the $V_{DDA}$ supply by comparing it to a threshold selected by the ALS[1:0] bits in the PWR control register 1 (PWR_CR1).

The AVD is enabled by setting the AVDEN bit in PWR control register 1 (PWR_CR1).

An AVDO flag is available in the PWR control status register 1 (PWR_CSR1) to indicate whether $V_{DDA}$ is higher or lower than the AVD threshold. This event is internally connected to the EXTI and can generate an interrupt if enabled through the EXTI registers. The AVDO interrupt can be generated when $V_{DDA}$ drops below the AVD threshold and/or when $V_{DDA}$ rises above the AVD threshold depending on EXTI rising/falling edge configuration. As an example the service routine could indicate when the $V_{DDA}$ supply drops below a minimum level.

**Figure 28. AVD thresholds**

1. For thresholds and hysteresis values, refer to the datasheets.
6.5.5 Battery voltage thresholds

In V_{BAT} mode, the battery voltage supply can be monitored by comparing it with two threshold levels: V_{BAT_{high}} and V_{BAT_{low}}. The V_{BAT} supply monitoring can be enabled/disabled via MONEN bit in *PWR control register 2 (PWR_CR2)*. When it is enabled, the battery voltage thresholds increase power consumption.

VBATH and VBATL wakeup interrupts are available on the RTC tamper signals (see *Section 50: Real-time clock (RTC)*).

Figure 29. VBAT thresholds

1. For thresholds and hysteresis values, refer to the datasheets.
6.5.6 **Temperature thresholds**

A dedicated temperature sensor cell is embedded in the power control. The junction temperature can be monitored by comparing it with two threshold levels, TEMP\textsubscript{high} and TEMP\textsubscript{low}. TEMPH and TEMPL flags in the **PWR control register 2 (PWR\_CR2)**, indicate whether the device temperature is higher or lower than the threshold. The temperature monitoring can be enabled/disabled via MONEN bit in **PWR control register 2 (PWR\_CR2)**. When enabled, the temperature thresholds increase power consumption. As an example the levels may be used to trigger a routine to perform temperature control tasks.

TEMPH and TEMPL wakeup interrupts are available on the RTC tamper signals (see Section 50: Real-time clock (RTC)).

Figure 30. Temperature thresholds

![Temperature thresholds diagram](MSv40345V1)

1. For thresholds and hysteresis values, refer to the datasheets.

6.5.7 **V\textsubscript{CORE} maximum voltage level detector**

V\textsubscript{CORE} is protected against too high voltages in the direct SMPS configuration. V\textsubscript{CORE} overvoltage protection is enabled at startup by hardware once the SMPS configuration has been programmed into **PWR control register 3 (PWR\_CR3)**. The two following configurations exist:

- **V\textsubscript{CORE} voltage level stays within range:**
  - The ACTVOSRDY bit in **PWR control status register 1 (PWR\_CSR1)** indicates valid voltage levels.
  - The system operates normally and V\textsubscript{CORE} overvoltage protection is disabled.

- **V\textsubscript{CORE} overvoltages (due to a wrongly programmed SMPS):**
  - The hardware forces the SMPS voltage level to 1.0 V.
  - ACTVOSRDY indicates invalid voltage levels. In this case the software must be corrected and re-loaded to program a correct SMPS configuration that matches the application supply connections. The system must be power cycled.
6.6 Power management

The power management block controls the $V_{\text{CORE}}$ supply in accordance with the system operation modes (see Section 6.6.1).

The $V_{\text{CORE}}$ domain is split into the following power domains.

- CPU domain (CD) containing some peripherals and the Cortex®-M7 Core (CPU)
- SmartRun domain (SRD) containing some peripherals and the system control

The CPU and SmartRun domains can operate in one of the following operating modes:

- DRun/Run/Run* (power ON, clock ON)
- DStop2/DStop/Stop (power ON, clock OFF)
- Standby (power OFF, clock OFF).

The CPU domain is a common power domain for the CPU and AXI peripherals on one hand, and AHB masters and peripherals outside the SmartRun domain on the other hand.

However, system SmartRun domain power modes depend on the CPU domain modes.

For the system SmartRun domain to operate in Stop mode, the CPU domain must be in DStop or DStop2 mode.

The CPU and system SmartRun domains are supplied from a single regulator at a common $V_{\text{CORE}}$ level. The $V_{\text{CORE}}$ supply level follows the system operating mode (Run, Stop, Standby). The CPU domain can be set in a specific retention level, known as DStop2, whereby the logic is switched off and the register contents are retained. Selection between Dstop or DStop2 is made through the RETDS_CD bit of $\text{PWR CPU control register (PWR\_CPUCR)}$. 
The content of all memories is retained in DStop and DStop2. Further power saving can be made by selectively switching off individual memory blocks. This is done by means of bits xxxSO bits of PWR control register 1 (PWR_CR1).

The following voltage scaling features allow controlling the power with respect to the required system performance (see Section 6.6.2: Voltage scaling):

- To obtain a given system performance, the corresponding voltage scaling must be set in accordance with the system clock frequency. To do this, configure the VOS bits to get the Run mode voltage scaling.
- To obtain the best trade-off between power consumption and exit-from-Stop mode latency, configure the SVOS bits to get Stop mode voltage scaling.

6.6.1 Operating modes

Several system operating modes are available to tune the system according to the performance required, means when the CPU does not need to execute code and is waiting for an external event. It is up to the user to select the operating mode that gives the best compromise between low power consumption, short startup time and available wakeup sources.

The operating modes allow controlling the clock distribution to the different system blocks and powering them. The system operating mode is driven by the CPU subsystem and system SmartRun autonomous wakeup. The CPU subsystem can span different configurations depending on its peripheral allocation (see Section 8.5.11: Peripheral clock gating control).

The operating modes described below are available for the different system blocks (see Table 33).

CPU subsystem modes

- **CRun**
  CPU and CPU subsystem peripheral(s) allocated via PERxEN bits in the RCC registers, are clocked.

- **CSleep**
  The CPU clock is stalled and the CPU subsystem allocated peripheral(s) clock operates according to PERxLPEN bits in the RCC registers.

- **CStop**
  CPU and CPU subsystem peripheral(s) clocks are stalled. When the CPU subsystem is in CStop mode, the CPU domain is either in DStop or Dstop2.

CPU domain modes

- **DRun**
  The domain bus matrix is clocked. The CPU subsystem operates in CRun or CSleep mode.

- **DStop**
  The domain bus matrix clock is stalled. The CPU subsystem operates in CStop mode and the RETDS_CD bit of PWR CPU control register (PWR_CPUCR) selects the
DStop mode.
The CPU domain peripherals able to operate in Stop mode are still operational.

- **DStop2**
The domain bus matrix clock is stalled. The CPU subsystem is in CStop mode and the RETDS_CD bit of _PWR CPU control register (PWR_CPUCR)_ selects the DStop2 mode.
The CPU domain peripherals able to operate in Stop mode are no longer operational.
In other words, no peripherals in CPU domain are operational.

The main differences between DStop and DStop2 are given below:
- In DStop mode, the entire logic is still supplied.
- In DStop2 mode, memories and registers are maintained, while asynchronous logic is switched off. This allows further leakage current reduction. Upon exiting DStop2, the CPU domain can resume normal execution.

The system state is retained in DStop and DStop2.

**System/SmartRun domain modes**

- **Run/Run***
The system clock and the SmartRun domain bus matrix clock are running.
  - The CPU subsystem is in CRun or CSleep mode
    or
  - A wakeup signal is active (system SmartRun Autonomous mode).
The Run* mode is entered after a POR reset and a wakeup from Standby. In Run* mode the performance is limited and the system supply configuration must be programmed in _PWR control register 3 (PWR_CR3)_ . The system enters Run mode only when the ACTVOSRDY bit in _PWR control status register 1 (PWR_CSR1)_ is set to 1.

- **Stop**
The system clock and the SmartRun domain bus matrix clock are stalled.
  - The CPU domain is in DStop or DStop2 mode
    and
  - The CPU subsystem is in CStop
    and
  - All wakeup signals are inactive
    and
  - The PDDS_SRD bit of _PWR CPU control register (PWR_CPUCR)_ selects the Stop mode

- **Standby**
The system is powered down.
  - The CPU domain and CPU subsystem are powered off
    and
  - All wakeup signals are inactive
    and
  - The PDDS_SRD bit of _PWR CPU control register (PWR_CPUCR)_ selects the Standby mode.
In Run mode, the power consumption can be reduced by one of the following means:

- lowering the system performance by slowing down the system clocks and reducing the V$_{CORE}$ supply level through VOS voltage scaling bits
- gating the clocks to the APBx and AHBx peripherals when they are not used, through PERxEN bits.

### Table 33. Low-power mode summary

<table>
<thead>
<tr>
<th>System</th>
<th>SRD domain</th>
<th>CPU domain</th>
<th>CPU</th>
<th>Entry</th>
<th>Wakeup</th>
<th>Sys-oscillator</th>
<th>System clk</th>
<th>Domain bus matrix clk</th>
<th>Peripheral clk</th>
<th>CPU clk</th>
<th>Voltage regulator</th>
<th>Domain supply</th>
<th>Voltage scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>SRDRun</td>
<td></td>
<td>CRun</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SRDRun</td>
<td></td>
<td>CSleep</td>
<td>WFI or return from ISR or WFE</td>
<td>Any interrupt or event</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Autono- mous</td>
<td></td>
<td>DStop$^2$</td>
<td>CStop</td>
<td>SLEEPDEEP bit + WFI or return from ISR or WFE</td>
<td>Any EXTI interrupt or event</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop$^5$</td>
<td>SRDStop</td>
<td></td>
<td>DStop$^2$</td>
<td>SLEEPDEEP bit + WFI or return from ISR or WFE or wakeup source cleared$^6$</td>
<td>WKUP pins rising or falling edge, RTC alarm (Alarm A or Alarm B), RTC wakeup event, RTC tamper events, RTC timestamp event, external reset in NRST pin, IWDG reset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. The CPU subsystem peripherals having a PERxLPEN bit, operate accordingly.
2. The CPU subsystem is in CStop and RETDS_CD selects DStop.
3. SmartRun domain peripherals having a PERxAMEN bit, operate accordingly.
4. The CPU subsystem is in CStop and RETDS_CD selects DStop2.
5. The CPU domain needs to be in DStop or DStop2 mode, no wakeup signal is active in SmartRun domain and PDDS_SRD bit selects the Stop mode.
6. When the CPU is in CStop and SmartRun domain in Autonomous mode, the last EXTI wakeup source is cleared.
7. When the system oscillator HSI or CSI is used, the state is controlled by HSIKERON and CSIKERON, otherwise the system oscillator is OFF.
6.6.2 Voltage scaling

The CPU and SmartRun domains are supplied from a single voltage regulator supporting voltage scaling with the following features:

- **Run mode voltage scaling**
  - VOS0: scale 0
  - VOS1: scale 1
  - VOS2: scale 2
  - VOS3: scale 3

- **Stop mode voltage scaling**
  - SVOS3: scale 3
  - LP-SVOS4: scale 4
  - LP-SVOS5: scale 5

For more details on voltage scaling values, refer to the product datasheets.

After reset, the system starts on the lowest Run mode voltage scaling (VOS3). The voltage scaling can then be changed on-the-fly by software by programming VOS bits in the PWR SmartRun domain control register (PWR_SRDCR) according to the required system performance. When exiting from Stop mode or Standby mode, the Run mode voltage scaling is reset to the default VOS3 value.

Before entering Stop mode, the software can preselect the SVOS level in PWR control register 1 (PWR_CR1). The Stop mode voltage scaling for SVOS4 and SVOS5 also sets the voltage regulator in Low-power (LP) mode to further reduce power consumption. When preselecting SVOS3, the use of the voltage regulator low-power mode (LP) can be selected by LPDS bit in PWR control register 1 (PWR_CR1).
6.6.3 Power control modes

The power control block handles the $V_{\text{CORE}}$ supply for system Run, Stop and Standby modes.

The system operating mode depends on the CPU subsystem modes (CRun, CSleep, CStop), on the domain modes (DRun, DStop, DStop2) and on the system SmartRun autonomous wakeup:

- In Run mode, $V_{\text{CORE}}$ is defined by the VOS voltage scaling.
  The CPU subsystem is either in CRun or CSleep, or an EXTI wakeup is active.

- In Stop mode, $V_{\text{CORE}}$ is defined by the SVOS voltage scaling.
  The CPU subsystem is in CStop mode and all EXTI wakeups are inactive. The CPU domain is either in DStop or DStop2 mode.

- In Standby mode, $V_{\text{CORE}}$ supply is switched off.
  The CPU domain and CPU subsystem are powered off and all EXTI wakeups are inactive.

The CPU domain mode selection between DStop and DStop2 is configured via RETDS_CD bit in PWR CPU control register (PWR_CPUCR). The system/SmartRun mode selection between Stop and Standby is configured via PDDS_SRD bit in PWR CPU control register (PWR_CPUCR).

The system enters Standby when PDDS_SRD bit allows it and stays otherwise in Stop mode (CPU domain in DStop or Dstop2).

Table 34 depicts all possible low-power mode states.
Table 34. PDDS_SRD and RETDS_CD low-power mode control

<table>
<thead>
<tr>
<th>PWR_CPUCCR</th>
<th>RETDS_CD</th>
<th>PDDS_SRD</th>
<th>CPU domain mode</th>
<th>SmartRun domain mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>DStop</td>
<td>Run or Stop</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>Standby</td>
<td>Standby</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 33. Power-control modes detailed state diagram
After a system reset, the CPU is in CRun mode.

Power control state transitions are initiated by the following events:

- The CPU goes to CStop mode or wakes up from CStop mode (state transitions in Run mode are marked in green).
- The system enters or exits Stop mode (state transitions marked in blue)
  - blue transitions: The system enters Stop mode with the CPU domain in DStop or Dstop2 and the CPU in CStop. The system wakes up from Stop. The CPU and CPU domain are restarted. When exiting system Stop mode, the STOPF bit is set.
  - yellow transitions: The system toggles between Run and Stop mode with the CPU domain in DStop or Dstop2 without waking up the CPU (Autonomous mode).
- The system enters or exits from Standby mode (state transitions are marked in pink).
  - When exiting from Standby mode, the SBF bit is set.

Table 35 shows the flags indicating from which mode the domain/system exits. The CPU features a set of flags that can be read from `PWR CPU control register (PWR_CPUCR)`.

<table>
<thead>
<tr>
<th>System mode</th>
<th>CPU domain mode</th>
<th>SBF</th>
<th>STOPF</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>DRun</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Run</td>
<td>DStop or DStop2</td>
<td>0</td>
<td>0</td>
<td>CPU domain and system contents retained</td>
</tr>
<tr>
<td>Stop</td>
<td>DStop or DStop2</td>
<td>0</td>
<td>1</td>
<td>CPU domain and system contents retained, clock system reset.</td>
</tr>
<tr>
<td>Standby</td>
<td>OFF</td>
<td>1</td>
<td>0</td>
<td>CPU domain and system contents lost</td>
</tr>
</tbody>
</table>

### 6.6.4 Power management examples

*Figure 34* shows $V_{\text{CORE}}$ voltage scaling behavior in Run mode.

*Figure 35* shows $V_{\text{CORE}}$ voltage scaling behavior in Stop mode.

*Figure 36* shows $V_{\text{CORE}}$ voltage regulator and voltage scaling behavior in Standby mode.

*Figure 37* shows $V_{\text{CORE}}$ voltage scaling behavior in Run mode with CPU domain in DStop or Dstop2 mode.

**Example of $V_{\text{CORE}}$ voltage scaling behavior in Run mode**

*Figure 34* illustrates the following system operation sequence example:

1. After reset, the system starts from HSI with VOS3.
2. The system performance is first increased to a medium-speed clock from the PLL with voltage scaling VOS2. To do this:
   a) Program the voltage scaling to VOS2.
   b) Once the $V_{\text{CORE}}$ supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.
3. The system performance is then increased to high-speed clock from the PLL with voltage scaling VOS1. To do this:
   a) Program the voltage scaling to VOS1.
b) Once the $V_{\text{CORE}}$ supply has reached the required level indicated by VOSRDY, increase the clock frequency.

4. The system performance is then reduced to a medium-speed clock with voltage scaling VOS2. To do this:
   a) First decrease the system frequency.
   b) Then decrease the voltage scaling to VOS2.

5. The next step is to reduce the system performance to HSI clock with voltage scaling VOS3. To do this:
   a) Switch the clock to HSI.
   b) Disable the PLL.
   c) Decrease the voltage scaling to VOS3.

6. The system performance can then be increased to high-speed clock from the PLL. To do this:
   a) Program the voltage scaling to VOS1.
   b) Once the $V_{\text{CORE}}$ supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.

When the system performance (clock frequency) is changed, VOS must be set accordingly; otherwise the system might be unreliable.

**Figure 34. Dynamic voltage scaling in Run mode**
Example of $V_{\text{CORE}}$ voltage scaling behavior in Stop mode

*Figure 35* illustrates the following system operation sequence example:

1. The system is running from the PLL in high-performance mode ($V_{\text{OS1}}$ voltage scaling).
2. The CPU subsystem deallocates all peripherals related to AHB1 and AHB2. As a consequence, the clocks $hclk[2:1]$ and the related APB1 and APB2 peripherals clocks ($pclk1$ and $pclk2$) are stopped. The system still provides the high-performance system clock, hence the voltage scaling must stay at $V_{\text{OS1}}$ level.
3. In a second step, the CPU subsystem enters CStop mode, the CPU domain enters DStop or DStop2 mode (Dstop or Dstop2 is selected by RETDS_CD bit in *PWR CPU control register (PWR_CPUCR)*) and the system enters Stop mode. The system clock is stopped and the hardware lowers the voltage scaling to the software preselected SVOS4 level.
4. The CPU subsystem is then woken up. The system exits Stop mode, the CPU domain exits DStop or DStop2 mode and the CPU subsystem exits CStop mode. The hardware then sets the voltage scaling to $V_{\text{OS3}}$ level and waits for the requested supply level to be reached before enabling the HSI clock. Once the HSI clock is stable, the system clock, the CPU domain clock and the CPU subsystem clock are enabled. Several system clock cycles are needed before the bus clocks are activated according to the xxxEN bits in the RCC.
5. The CPU subsystem allocates a peripheral in the APB1/APB2 domain. The related $hclk$ and $pclk$ peripheral clocks are enabled.
6. The system performance is then increased. To do this:
   a) The software first sets the voltage scaling to $V_{\text{OS1}}$.
   b) Once the $V_{\text{CORE}}$ supply has reached the required level indicated by $V_{\text{OSRDY}}$, the clock frequency can be increased by enabling the PLL.
   c) Once the PLL is locked, the system clock can be switched.
Figure 35. Dynamic voltage scaling behavior with CPU domain and system in Stop mode

- VOS 1
- VOS 2
- (S)VOS 3
- SVOS 4
- VOS
- SVOS
- VOSRDY
- exti_c_wkup
- pwr_srd_wkup
- pwr_cd_wkup
- PLLxON
- ck_sys
- ck_fclk
- ck_hclk1/2/3
- clock stopped

Legend:
- CPU Domain RUN
- SmartRun RUN
- STOP
- wait
- Clock Stopped
- CPU Domain RUN
- SmartRun RUN
- Wait
- VOSRDY
- Run from PLL
- Run from HSI
- Run from PLL

Note: The figure illustrates the dynamic voltage scaling behavior with CPU domain and system in Stop mode, showing the transitions between different voltage levels and clock states.
Example of \( V_{\text{CORE}} \) voltage regulator/voltage scaling behavior in Standby mode

*Figure 36* illustrates the following system operation sequence example:

1. The system is running from the PLL in high-performance mode (VOS1 voltage scaling).
2. The CPU subsystem deallocates all peripherals related to AHB1 and AHB2. As a consequence the clocks hclk[2:1] and the related APB1 and APB2 peripherals clocks (pclk1 and pclk2) are stopped. The system performance is unchanged hence the voltage scaling does not change.
3. The CPU subsystem and the system enters Standby mode (selection through PDDS_SRD bit). The system clock is stopped and the voltage regulator is switched off.
4. The system is then woken up by a wakeup source. The system exits Standby mode. The hardware sets the voltage scaling to the default VOS3 level and waits for the requested supply level to be reached before enabling the default HSI oscillator. Once the HSI clock is stable, the system clock and the CPU subsystem clock are enabled. The software must then check the ACTVOSRDY is valid before changing the system performance.
5. The CPU subsystem allocates a peripheral in the APB1/APB2 domain. The related hclk and pclk peripheral clocks are enabled.
6. In a next step, increase the system performance. To do this:
   a) The software first increases the voltage scaling to VOS1 level.
   b) Before enabling the PLL, the software waits for the requested supply level to be reached by monitoring VOSRDY bit.
   c) Once the PLL is locked, the system clock can be switched.
Figure 36. Dynamic voltage scaling system Standby mode
Example of $V_{\text{CORE}}$ voltage scaling behavior in Run mode with CPU domain in DStop or DStop2 mode

*Figure 37* illustrates the following system operation sequence example:

1. The system is running from the PLL with system in high-performance mode (VOS1 voltage scaling).

2. The CPU subsystem deallocates all peripherals related to AHB1 and AHB2. As a consequence the clocks hclk[2:1] and the related APB1 and APB2 peripherals clocks (pclk1 and pclk2) are stopped. The system performance is unchanged hence the voltage scaling does not change.

3. The CPU subsystem then enters CStop mode and the CPU domain enters DStop or DStop2 mode (selected by RETDS_CD bit in `PWR CPU control register` (PWR_CPUCR)). The CPU AXI bus matrix clock is stopped. At the same time the system/SmartRun domain enters Stop mode. The system clock is stopped and the hardware lowers the voltage scaling to the software preselected SVOS4 level.

4. The system is then woken up by a SmartRun autonomous mode wakeup event. The system exits Stop mode. The hardware sets the voltage scaling to the default VOS3 level and waits for the requested supply level to be reached before enabling the HSI clock. Once the HSI clock is stable, the system clock is enabled. The system is running in SmartRun Autonomous mode.

5. The SmartRun autonomous mode wakeup source is then cleared, causing the system to enter Stop mode. The system clock is stopped and the voltage scaling is lowered to the software preselected SVOS4 level.

6. The CPU subsystem is then woken up. The system exits Stop mode, the CPU domain exits DStop/DStop2 mode and the CPU subsystem exits CStop mode. The hardware sets the voltage scaling to the default VOS3 level and waits for the requested supply level to be reached before enabling the default HSI oscillator. Once the HSI clock is stable, the system clock and the CPU subsystem clock are enabled.
Figure 37. Dynamic voltage scaling behavior with CPU domain in DStop or DStop2 mode and SmartRun domain in Autonomous mode
### 6.7 Low-power modes

Several low-power modes are available to save power when the CPU does not need to execute code (when waiting for an external event). It is up to the user application to select the mode that gives the best compromise between low power consumption, short startup time and available wakeup sources:

- Slowing down system clocks (see *Section 8.5.6: System clock (sys_ck)*)
- Controlling individual peripheral clocks (see *Section 8.5.11: Peripheral clock gating control*)
- Low-power modes
  - CSleep (CPU clock stopped)
  - CStop (CPU subsystem clock stopped). Whenever the CPU subsystem enters CStop, the CPU domain jointly enters a low-power state:
    a) DStop (power domain clocks stopped, retention) or DStop2 (power domain clocks stopped, deep retention). Memories low-power mode for further optimization (Flash memory low-power mode or RAM shut-off)
    b) Standby, along with the whole system/SmartRun domain (see *Figure 33* for the state transition diagram)
  - Stop (system clock stopped)
  - Standby (system powered down)

#### 6.7.1 Slowing down system clocks

In Run mode the speed of the system clock ck_sys can be reduced. For more details refer to *Section 8.5.6: System clock (sys_ck)*.

#### 6.7.2 Controlling peripheral clocks

In Run mode, the HCLKx and PCLKx for individual peripherals can be stopped by configuring at any time PERxEN bits in RCC_C1_xxxxENR or RCC_DnxxxxENR to reduce power consumption.

To reduce power consumption in CSleep mode, the individual peripheral clocks can be disabled by configuring PERxLPEN bits in RCC_C1_xxxxLPENR or RCC_DnxxxxLPENR. For the peripherals still receiving a clock in CSleep mode, their clock can be slowed down before entering CSleep mode and clock gating can be enabled (RCC_CKGAENR register).

#### 6.7.3 Entering low-power modes

CPU subsystem CSleep and CStop low-power modes are entered by the MCU when executing the WFI (wait for interrupt) or WFE (wait for event) instructions, or when the SLEEPONEXIT bit in the Cortex®-M system control register is set on Return from ISR.

The CPU domain either enters DStop or DStop2 low-power mode when the CPU subsystem enters CStop mode, or enters Standby mode together with the system/SmartRun domain.

The system can enter Stop or Standby low-power mode when all EXTI wakeup sources are cleared (see *Figure 33*).
6.7.4 Exiting from low-power modes

The CPU subsystem exits CSleep mode through any interrupt or event depending on how the low-power mode was entered:

- If the WFI instruction or Return from ISR was used to enter to low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the system.
- If the WFE instruction is used to enter to low-power mode, the CPU exits low-power mode as soon as an event occurs. The wakeup event can be generated by one of the followings:
  - an NVIC IRQ interrupt
    
    When SEVONPEND = 0 in the Cortex®-M7 system control register, the interrupt must be enabled in the peripheral control register and in the NVIC.
    
    When the MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared. Only NVIC interrupts with sufficient priority wake up and interrupt the MCU.

    When SEVONPEND = 1 in the Cortex®-M7 system control register, the interrupt must be enabled in the peripheral control register and optionally in the NVIC.
    
    When the MCU resumes from WFE, the peripheral interrupt pending bit and, when enabled, the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
    
    All NVIC interrupts wake up the MCU, even the disabled ones.
    
    Only enabled NVIC interrupts with sufficient priority wake up and interrupt the MCU.
  - an event
    
    An EXTI line must be configured in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit, as the pending bits corresponding to the event line are not set. It might be necessary to clear the interrupt flag in the peripheral.

The CPU subsystem exits from CStop, DStop and Stop modes by enabling an EXTI interrupt or event depending on how the low-power mode was entered (see above).

The system can wake up from Stop mode by enabling an EXTI wakeup, without waking up the CPU subsystem. In this case the system operates in SmartRun Autonomous mode.

The CPU subsystem exits from Standby mode by enabling an external reset (NRST pin), an IWDG reset, a rising edge on one of the enabled WKUPx pins or a RTC event. Program execution restarts in the same way as after a system reset (such as boot pin sampling, option bytes loading or reset vector fetched).

6.7.5 CSleep mode

The CSleep mode applies only to the CPU subsystem. In CSleep mode, the CPU clock is stopped. The CPU subsystem peripheral clocks operate according to the values of PERxLPEN bits in RCC_C1_xxxxENR or RCC_DnxxxxENR.

Entering CSleep mode

The CSleep mode is entered according to Section 6.7.3: Entering low-power modes, when the SLEEPDEEP bit in the Cortex®-M system control register is cleared.

Refer to Table 36 for details on how to enter CSleep mode.
Exiting from CSleep mode

The CSleep mode is exited according to Section 6.7.4: Exiting from low-power modes. Refer to Table 36 for more details on how to exit from CSleep mode.

Table 36. CSleep mode

<table>
<thead>
<tr>
<th>CSleep mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode entry</td>
<td>WFI (wait for interrupt) or WFE (wait for event) while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP = 0 (refer to the Cortex®-M system control register)</td>
</tr>
<tr>
<td></td>
<td>– CPU NVIC interrupts and events cleared</td>
</tr>
<tr>
<td></td>
<td>On Return from ISR while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP = 0 and SLEEPONEXIT = 1 (refer to the Cortex®-M system control register.)</td>
</tr>
<tr>
<td></td>
<td>– CPU NVIC interrupts and events cleared</td>
</tr>
<tr>
<td>Mode exit</td>
<td>If WFI or Return from ISR was used for entry:</td>
</tr>
<tr>
<td></td>
<td>– any interrupt enabled in NVIC (refer to Table 123: NVIC)</td>
</tr>
<tr>
<td></td>
<td>If WFE was used for entry and SEVONPEND = 0:</td>
</tr>
<tr>
<td></td>
<td>– any event (refer to Section 20.5.3: EXTI CPU wakeup procedure)</td>
</tr>
<tr>
<td></td>
<td>If WFE was used for entry and SEVONPEND = 1:</td>
</tr>
<tr>
<td></td>
<td>– any interrupt even when disabled in NVIC (refer to Table 123: NVIC) or any event (refer to Section 20.5.3: EXTI CPU wakeup procedure)</td>
</tr>
<tr>
<td>Wakeup latency</td>
<td>None</td>
</tr>
</tbody>
</table>

6.7.6 CStop mode

The CStop mode applies only to the CPU subsystem. In CStop mode, the CPU clock is stopped.

When the system is in Run mode and the CPU subsystem in CStop mode, most peripheral clocks are stopped too. Only the peripherals having a PERxAMEN bit, operate accordingly and, if relevant, can request a kernel clock.

Entering CStop mode

The CStop mode is entered according to Section 6.7.3: Entering low-power modes, when the SLEEPDEEP bit in the Cortex®-M system control register is set. Refer to Table 37 for details on how to enter to CStop mode.

Exiting from CStop mode

The CStop mode is exited according to Section 6.7.4: Exiting from low-power modes. Refer to Table 37 for more details on how to exit from CStop mode.
6.7.7 DStop and DStop2 modes

Whenever the CPU subsystem enters CStop mode, the CPU domain either enters DStop or DStop2 mode, or goes into Standby mode together with the system/SmartRun domain (selection made by PDDS_SRD bit in PWR CPU control register (PWR_CPUCR)).

In DStop or DStop2 mode, the CPU clock is stopped (because the CPU is in CStop), the CPU domain bus clocks are stopped and most peripherals clocks are stopped too. Peripherals using the LSI or LSE clock and peripherals having a kernel clock are still able to operate, but only in SVOS3 mode.

The Flash memory can enter low-power Stop mode when it is enabled through FLPS bit in PWR control status register 1 (PWR_CSR1). The wakeup time to restart from the Flash memory in low-power mode is slightly longer.

It is mandatory to configure the Flash memory in low-power mode before entering DStop2 mode.

### Table 37. CStop mode

<table>
<thead>
<tr>
<th>CStop mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode entry</td>
<td>WiFi (wait for interrupt) or WFE (wait for event) while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP = 1 (refer to the Cortex®-M system control register)</td>
</tr>
<tr>
<td></td>
<td>– CPU NVIC interrupts and events cleared</td>
</tr>
<tr>
<td></td>
<td>– All CPU EXTI wakeup sources are cleared.</td>
</tr>
<tr>
<td></td>
<td>On Return from ISR while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP = 1 and SLEEPONEXIT = 1 (refer to the Cortex®-M system control register)</td>
</tr>
<tr>
<td></td>
<td>– CPU NVIC interrupts and events cleared</td>
</tr>
<tr>
<td></td>
<td>– All CPU EXTI wakeup sources are cleared.</td>
</tr>
</tbody>
</table>

| Mode exit | If WFI or Return from ISR was used for entry: |
|          | – EXTI Interrupt enabled in NVIC (refer to Table 123: NVIC) for peripherals not stopped or powered down |
|          | If WFE was used for entry and SEVONPEND = 0: |
|          | – EXTI event (refer to Section 20.5.3: EXTI CPU wakeup procedure) for peripherals not stopped or powered down |
|          | If WFE was used for entry and SEVONPEND = 1: |
|          | – EXTI interrupt even when disabled in NVIC (refer to Table 123: NVIC) or EXTI event (refer to Section 20.5.3: EXTI CPU wakeup procedure) for peripherals not stopped or powered down |

| Wakeup latency | EXTI and RCC wakeup synchronization (see Section 8.4.7: Power-on and wakeup sequences) |

### Table 38. DStop mode overview

<table>
<thead>
<tr>
<th>CPU</th>
<th>CPU domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRun or CSleep</td>
<td>DRun</td>
</tr>
<tr>
<td>CStop</td>
<td>DStop/DStop2</td>
</tr>
</tbody>
</table>
Entering DStop or DStop2 mode

The DStop (DStop2) mode is entered according to Section 6.7.3: Entering low-power modes, when RETDS_CD bit in PWR CPU control register (PWR_CPUCR) for the CPU domain, selects Stop (respectively Stop2).

Refer to Table 39 for details on how to enter DStop or DStop2 mode.

If the Flash memory programming is ongoing, the DStop/DStop2 mode entry is delayed until the memory access is finished.

If an access to the domain bus matrix is ongoing, the DStop/DStop2 mode entry is delayed until the domain bus access is complete.

Warning: The user must ensure that no allocated peripheral in the CPU domain has an active kernel clock, or are still clocked by LSI, LSE, HSI or CSI. The Flash memory must be configured in low-power mode (FLPS bit set in PWR_CR1 register) before entering DStop2.

Exiting from DStop or DStop2 mode

The DStop or DStop2 mode is exited according to Section 6.7.4: Exiting from low-power modes.

Refer to Table 39 for more details on how to exit DStop or DStop2 mode.

When exiting from DStop or DStop2 mode, the CPU subsystem clocks, domain(s) bus matrix clocks and voltage scaling depend on the system mode.

- When the system did not enter Stop mode, the CPU subsystem clocks, domain(s) bus matrix clocks and voltage scaling values are the same as when entering DStop or DStop2 mode.
- When the system has entered Stop mode, the CPU subsystem clocks, domain(s) bus matrix clocks and voltage scaling are reset.

Table 39. DStop or DStop2 mode

<table>
<thead>
<tr>
<th>DStop or DStop2 mode</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Mode entry**       | – The CPU subsystem enters CStop.  
– The RETDS_CD bit selects Stop --> Entry into DStop.  
– The RETDS_CD bit selects Stop2 --> Entry into DStop2 (no CPU peripherals with kernel clocks or cadenced by LSI, LSE, HSI or CSI). |
| **Mode exit**        | The CPU subsystem exits from CStop mode (see Table 37). |
| **Wakeup latency**   | EXTI and RCC wakeup synchronization (see Section 8.4.7: Power-on and wakeup sequences). |
6.7.8 Stop mode

The system SmartRun domain enters Stop mode only when the CPU subsystem is in CStop mode and the EXTI wakeup sources are inactive. In Stop mode, the system clock including a PLL and the SmartRun domain bus matrix clocks are stopped. When HSI or CSI is selected, the system oscillator operates according to the HSIKERON and CSIKERON bits in RCC_CR register. Other system oscillator sources are stopped.

In system SmartRun domain Stop mode, the CPU domain is in DStop or DStop2 mode.

In system Stop mode, the following features can be selected to remain active by programming individual control bits:

- Independent watchdog (IWDG)
  The IWDG is started by writing to its key register or by hardware option. Once started it cannot be stopped except by a reset (see Section 49: Independent watchdog (IWDG)).

- Real-time clock (RTC)
  This is configured via the RTCEN bit in the RCC Backup Domain Control Register (RCC_BDCR).

- Internal RC oscillator (LSI RC)
  This is configured via the LSION bit in the RCC Clock Control and Status Register (RCC_CSR).

- External 32.768 kHz oscillator (LSE OSC)
  This is configured via the LSEON bit in the RCC Backup Domain Control Register (RCC_BDCR).

- Peripherals in SmartRun domain capable of running on the LSI or LSE clock

- Peripherals in SmartRun domain having a kernel clock request
  In this case the PERxAMEN bit of these peripherals must be set to request the kernel clock (in the RCC_SRDAMR register)

- Peripherals in CPU domain capable of running on the LSI or LSE clock, if CPU domain is in DStop mode (as opposed to DStop2)

- Peripherals in the CPU domain having a kernel clock request, if the CPU domain is in DStop mode (as opposed to DStop2)

- Internal RC oscillators (HSI and CSI)
  This is configured via the HSIKERON and CSIKERON bits in the RCC Clock Control and Status Register (RCC_CSR).

- The ADC or DAC can also consume power during Stop mode, unless they are disabled before entering this mode. To disable them, the ADON bit in the ADC_CR2 register and the ENx bit in the DAC_CR register must both be written to 0.

The selected SVOS4 and SVOS5 levels add an additional startup delay when exiting from system Stop mode (see Table 40). An extra latency is added upon entering and exiting DStop2 mode as well.
Memory shut-off

In DStop or DStop2 mode, the content of the memory blocks is maintained. Further power optimization can be obtained by switching off some memory blocks. This optimization implies loss of the memory content. The user can select which memory is discarded during Stop mode by means of xxSO bits in PWR control register 1 (PWR_CR1) as indicated in Table 42.

### Table 40. Stop mode operation

<table>
<thead>
<tr>
<th>SVOS</th>
<th>LPDS</th>
<th>Stop mode voltage regulator operation</th>
<th>Wake-up latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVOS3</td>
<td>0</td>
<td>Main</td>
<td>No additional wakeup time</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>LP</td>
<td>Voltage regulator wakeup time from Low-power mode</td>
</tr>
<tr>
<td>SVOS4 or SVOS5</td>
<td>x</td>
<td>LP</td>
<td>Voltage regulator wakeup time from Low-power mode + voltage level wakeup time for SVOS4 or SVOS5 level to VOS3 level</td>
</tr>
</tbody>
</table>

### Table 41. Memory shut-off block selection

<table>
<thead>
<tr>
<th>Selection bit</th>
<th>Shut-off block during Stop mode (DStop or DStop2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXIRAM1SO</td>
<td>AXI SRAM1 shut-off control</td>
</tr>
<tr>
<td>AXIRAM2SO</td>
<td>AXI SRAM2 shut-off control</td>
</tr>
<tr>
<td>AXIRAM3SO</td>
<td>AXI SRAM3 shut-off control</td>
</tr>
<tr>
<td>AHBRAM1SO</td>
<td>AHB SRAM1 shut-off control</td>
</tr>
<tr>
<td>AHBRAM2SO</td>
<td>AHB SRAM2 shut-off control</td>
</tr>
<tr>
<td>ITCMSO</td>
<td>ITCM and ETM memories shut-off control</td>
</tr>
<tr>
<td>GFXSO</td>
<td>GFXMMU and JPEG memory shut-off control</td>
</tr>
<tr>
<td>HSITFSO</td>
<td>High-speed interface USB and FDCAN memories shut-off control</td>
</tr>
<tr>
<td>SRDRAMSO</td>
<td>SmartRun AHB SRAM shut-off control</td>
</tr>
</tbody>
</table>

### Entering Stop mode

The Stop mode is entered according to Section 6.7.3: Entering low-power modes.

Refer to Table 42 for details on how to enter Stop mode.

If the Flash memory programming is ongoing, the Stop mode entry is delayed until the memory access is finished.

If an access to a bus matrix is ongoing, the Stop mode entry is delayed until the bus matrix access is finished.

---

**Warning:** To allow peripherals having a kernel clock request to operate in Stop mode, the system must use SVOS3 level.
Exiting from Stop mode

The Stop mode is exited according to Section 6.7.4: Exiting from low-power modes. Refer to Table 42 for more details on how to exit from Stop mode.

When exiting Stop mode, the system clock, the SmartRun domain bus matrix clocks and voltage scaling are reset.

STOPF status flag in PWR CPU control register (PWR_CPUCR) indicates that the system has exited Stop mode (see Table 35).

### Table 42. Stop mode

<table>
<thead>
<tr>
<th>Stop mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode entry</td>
<td>When the CPU is in CStop mode and there is no active EXTI wakeup source and RUN_SRD = 0 (in PWR_CPUCR register)</td>
</tr>
<tr>
<td>Mode exit</td>
<td>On a EXTI wakeup</td>
</tr>
<tr>
<td>Wakeup latency</td>
<td>System oscillator startup (when disabled)</td>
</tr>
<tr>
<td></td>
<td>+ EXTI and RCC wakeup synchronization</td>
</tr>
<tr>
<td></td>
<td>+ Voltage scaling refer to Table 40 (see Section 6.6.2: Voltage scaling)</td>
</tr>
<tr>
<td></td>
<td>+ Exiting DStop2</td>
</tr>
</tbody>
</table>

I/O states in Stop mode

The I/O pin configuration remains unchanged in Stop mode.
6.7.9 Standby mode

The Standby mode allows achieving the lowest power consumption. Like Stop mode, it is based on CPU subsystem CStop mode. However the V\textsubscript{CORE} supply regulator is powered off.

The system SmartRun domain enters Standby mode only when the CPU domain is in DStop or DStop2, no EXTI is pending and PDDS\textsubscript{SRD} bit selects Standby. When the system/SmartRun domain enters Standby mode, the voltage regulator is disabled. The complete V\textsubscript{CORE} domain is consequently powered off. The PLLs, HSI oscillator, CSI oscillator, HSI48 and HSE oscillator are also switched off. The content of SRAM and registers is lost except for Backup domain registers (RTC registers, RTC backup register and backup RAM), and Standby circuitry (see Section 6.4.4: Backup domain).

In system Standby mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG)
  
  The IWDG is started by programming its key register or by hardware option. Once started, it cannot be stopped except by a reset (see Section 49: Independent watchdog (IWDG)).

- Real-time clock (RTC)
  
  This is configured via the RTCEN bit in the Backup domain control register (RCC\_BDCR).

- Internal RC oscillator (LSI RC)
  
  This is configured by the LSION bit in the control/status register (RCC\_CSR).

- External 32.768 kHz oscillator (LSE OSC)
  
  This is configured by the LSEON bit in the Backup domain control register (RCC\_BDCR).

Entering Standby mode

The Standby mode is entered according to Section 6.7.3: Entering low-power modes, when the PDDS\_SRD bit requests Standby (in PWR CPU control register (PWR\_CPUCR)).

Refer to Table 44 for more details on how to enter Standby mode.

Exiting from Standby mode

The Standby mode is exited according to Section 6.7.4: Exiting from low-power modes.

Refer to Table 44 for more details on how to exit from Standby mode.

The system exits from Standby mode when an external reset (NRST pin), an IWDG reset, a WKUP pin event, a RTC alarm, a tamper event, or a timestamp event is detected. All registers are reset after waking up from Standby except for power control and status registers (PWR control register 2 (PWR\_CR2), PWR control register 3 (PWR\_CR3)), SBF bit in PWR CPU control register (PWR\_CPUCR), PWR wakeup flag register (PWR\_WKUPFR), and PWR wakeup enable and polarity register (PWR\_WKUPEPR).

After waking up from Standby mode, the program execution restarts in the same way as after a system reset (boot option sampling, boot vector reset fetched). The SBF status flag in PWR CPU control register (PWR\_CPUCR) indicates from which mode the system has exited (see Table 43).
I/O states in Standby mode

In Standby mode, all I/O pins are high impedance without pull, except for:

- Reset pad (still available)
- PC14 and PC15 pins if configured for LSE oscillator
- PC13 pin if configured as RTC_OUT1, RTC_TS, TAMP_IN1, TAMP_OUT2 or TAMP_OUT3, assuming they have been configured by the RTC or the TAMPER
- PI8 pin if configured as RTC_OUT2, TAMP_IN2 or TAMP_OUT3, assuming they have been configured by the RTC or the TAMPER
- PC1 pin if configured as TAMP_IN3, assuming it has been configured by the TAMPER
- WKUP pins (if enabled).

The WKUP pin pull configuration can be defined through WKUPPUPDx[1:0] bits in PWR wakeup enable and polarity register (PWR_WKUPEPR).

6.7.10 Monitoring low-power modes

The devices feature state monitoring pins that monitor the CPU and Domain state transitions to low-power mode (refer to Table 45 for the list of pins and their description). The GPIO pin corresponding to each monitoring signal has to be programmed in alternate function mode.

This feature is not available in Standby mode since these I/O pins are switched to high impedance.

Table 45. Overview of low-power mode monitoring pins

<table>
<thead>
<tr>
<th>Power state monitoring pins</th>
<th>Description</th>
<th>Pin assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSLEEP</td>
<td>CPU clock OFF</td>
<td>PC3</td>
</tr>
<tr>
<td>CSTOP</td>
<td>CPU domain in low-power mode</td>
<td>PC2</td>
</tr>
<tr>
<td>NDSTOP2</td>
<td>CPU domain retention mode selection</td>
<td>PA4</td>
</tr>
</tbody>
</table>
The state of the monitoring pins reflect the mode of the CPUs and domains. Refer to Table 46 for a description of the GPIO state depending on the CPU and domain state.

<table>
<thead>
<tr>
<th>CSLEEP</th>
<th>CSTOP</th>
<th>NDSTOP2</th>
<th>CD domain power state</th>
<th>SRD domain power state</th>
<th>CPU clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>DRun</td>
<td>SRDRun</td>
<td>ON</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>DRun</td>
<td>SRDRun</td>
<td>OFF</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>DStop</td>
<td>SRDRun or SRDStop</td>
<td>OFF</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>DStop2</td>
<td>SRDRun or SRDStop</td>
<td>OFF</td>
</tr>
</tbody>
</table>
6.8 **PWR registers**

The PWR registers can be accessed in word, half-word and byte format, unless otherwise specified.

### 6.8.1 PWR control register 1 (PWR_CR1)

Address offset: 0x000

Reset value: 0xF000 C000

<table>
<thead>
<tr>
<th>Bit 31:28</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

Bit 27 **SRDRAMSO**: SmartRun domain AHB memory shut-off in DStop/DStop2 low-power mode

0: SmartRun domain AHB memory content is kept in DStop or DStop2 mode  
1: SmartRun domain AHB memory content is lost in DStop or DStop2 mode

Bit 26 **HSITFSO**: high-speed interfaces USB and FDCAN memory shut-off in DStop/DStop2 mode

0: USB and FDCAN memories content is kept in DStop or DStop2 mode  
1: USB and FDCAN memories content is lost in DStop or DStop2 mode

Bit 25 **GFXSO**: GFXMMU and JPEG memory shut-off in DStop/DStop2 mode

0: GFXMMU and JPEG memory content is kept in DStop or DStop2 mode  
1: GFXMMU and JPEG memory content is lost in DStop or DStop2 mode

Bit 24 **ITCMSO**: Instruction TCM and ETM memory shut-off in DStop/DStop2 mode

0: ITCM and ETM memories content is kept in DStop or DStop2 mode  
1: ITCM and ETM memories content is lost in DStop or DStop2 mode

Bit 23 **AHBRAM2SO**: AHB SRAM2 shut-off in DStop/DStop2 mode

0: AHB SRAM2 content is kept in DStop or DStop2 mode  
1: AHB SRAM2 content is lost in DStop or DStop2 mode

Bit 22 **AHBRAM1SO**: AHB SRAM1 shut-off in DStop/DStop2 mode

0: AHB SRAM1 content is kept in DStop or DStop2 mode  
1: AHB SRAM1 content is lost in DStop or DStop2 mode

Bit 21 **AXIRAM3SO**: AXI SRAM3 shut-off in DStop/DStop2 mode

0: AXI SRAM3 content is kept in DStop or DStop2 mode  
1: AXI SRAM3 content is lost in DStop or DStop2 mode
Bit 20 **AXIRAM2SO**: AXI SRAM2 shut-off in DStop/DStop2 mode
0: AXI SRAM2 content is kept in DStop or DStop2 mode
1: AXI SRAM2 content is lost in DStop or DStop2 mode

Bit 19 **AXIRAM1SO**: AXI SRAM1 shut-off in DStop/DStop2 mode
0: AXI SRAM1 content is kept in DStop or DStop2 mode
1: AXI SRAM1 content is lost in DStop or DStop2 mode

Bits 18:17 **ALS[1:0]**: analog voltage detector level selection
These bits select the voltage threshold detected by the AVD.
00: 1.7 V
01: 2.1 V
10: 2.5 V
11: 2.8 V

Bit 16 **AVDEN**: peripheral voltage monitor on VDDA enable
0: peripheral voltage monitor on VDDA disabled
1: peripheral voltage monitor on VDDA enabled

Bits 15:14 **SVOS[1:0]**: system stop mode voltage scaling selection
These bits control the VCORE voltage level in system Stop mode, to obtain the best trade-off between power consumption and performance.
00: reserved
01: SVOS5 scale 5
10: SVOS4 scale 4
11: SVOS3 scale 3 (default)

Bit 13 **AVD_READY**: analog voltage ready
This bit is only used when the analog switch boost needs to be enabled (see BOOSTE bit).
It must be set by software when the expected VDDA analog supply level is available.
The correct analog supply level is indicated by the AVDO bit (PWR_CSR1 register) after setting the AVDEN bit and selecting the supply level to be monitored (ALS bits).
0: peripheral analog voltage VDDA not ready (default)
1: peripheral analog voltage VDDA ready

Bit 12 **BOOSTE**: analog switch VBoost control
This bit enables the booster to guarantee the analog switch AC performance when the VDD supply voltage is below 2.7 V (reduction of the total harmonic distortion to have the same switch performance over the full supply voltage range)
The VDD supply voltage can be monitored through the PVD and the PLS bits.
0: booster disabled (default)
1: booster enabled if analog voltage ready (AVD_READY = 1)

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **FLPS**: Flash memory Low-power mode in DStop or DStop2 mode
When it is set, the Flash memory enters Low-power mode when the CPU domain is in DStop/DStop2 mode.
The power consumption is improved with a slightly longer wakeup time.
**It is mandatory to set FLPS before entering DStop2 mode.**
0: Flash memory remains in normal mode when the CPU domain enters DStop (quick restart time).
1: Flash memory enters Low-power mode when the CPU domain enters DStop/DStop2 (low-power consumption).
Bit 8  **DBP**: disable Backup domain write protection

In reset state, the RCC_BDCR register, the RTC registers (including the backup registers), BREN and MOEN bits in PWR_CR2 register, are protected against parasitic write access. This bit must be set to enable write access to these registers.

- 0: access to RTC, RTC backup registers and backup SRAM disabled
- 1: access to RTC, RTC backup registers and backup SRAM enabled

Bits 7:5  **PLS[2:0]**: programmable voltage detector level selection

These bits select the voltage threshold detected by the PVD.

- 000: 1.95 V
- 001: 2.1 V
- 010: 2.25 V
- 011: 2.4 V
- 100: 2.55 V
- 101: 2.7 V
- 110: 2.85 V
- 111: PVD_IN pin

*Note: Refer to Section “Electrical characteristics” of the product datasheet for more details.*

Bit 4  **PVDE**: programmable voltage detector enable

- 0: programmable voltage detector disabled
- 1: programmable voltage detector enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0  **LPDS**: low-power Deepsleep with SVOS3 (SVOS4 and SVOS5 always use low-power, regardless of the setting of this bit)

- 0: voltage regulator or SMPS in Main mode (MR) when SVOS3 selects Stop
- 1: voltage regulator or SMPS in Low-power mode (LPR) when SVOS3 selects Stop

### 6.8.2 PWR control status register 1 (PWR_CSR1)

**Address offset**: 0x004

**Reset value**: 0x0000 4000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>MMCMDD</th>
<th>AVDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ACVTOS[1,0]</td>
<td>ACVTOS[1,0]</td>
<td>ACVTOS[1,0]</td>
<td>ACVTOS[1,0]</td>
<td>ACVTOS[1,0]</td>
<td>ACVTOS[1,0]</td>
<td>ACVTOS[1,0]</td>
<td>ACVTOS[1,0]</td>
<td>ACVTOS[1,0]</td>
<td>ACVTOS[1,0]</td>
<td>ACVTOS[1,0]</td>
<td>ACVTOS[1,0]</td>
<td>ACVTOS[1,0]</td>
<td>ACVTOS[1,0]</td>
<td>PVDD</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
Bits 31:18  Reserved, must be kept at reset value.

Bit 17  **MMCVDO**: voltage detector output on V\textsubscript{DDMMC}
This bit is set and cleared by hardware.
0: V\textsubscript{DDMMC} is below 1.2 V.
1: V\textsubscript{DDMMC} is above or equal to 1.2 V.

Bit 16  **AVDO**: analog voltage detector output on V\textsubscript{DDA}
This bit is set and cleared by hardware. It is valid only if AVD on V\textsubscript{DDA} is enabled by the AVDEN bit.
0: V\textsubscript{DDA} is equal or higher than the AVD threshold selected with the ALS[2:0] bits.
1: V\textsubscript{DDA} is lower than the AVD threshold selected with the ALS[2:0] bits.

*Note: Since the AVD is disabled in Standby mode, this bit is equal to 0 after Standby or reset until the AVDEN bit is set.*

Bits 15:14  **ACTVOS[1:0]**: VOS currently applied for V\textsubscript{CORE} voltage scaling selection.
These bits reflect the last VOS value applied to the voltage regulator.

Bit 13  **ACTVOSRDY**: voltage levels ready bit for currently used VOS
This bit is set to 1 by hardware when the voltage regulator and the SMPS are both disabled and Bypass mode is selected in PWR control register 3 (PWR_CR3).
0: voltage level invalid, above or below current VOS selected level
1: voltage level valid, at current VOS selected level

Bits 12:5  Reserved, must be kept at reset value.

Bit 4  **PVDO**: programmable voltage detect output
This bit is set and cleared by hardware. It is valid only if the PVD has been enabled by the PVDE bit.
0: V\textsubscript{DD} is equal or higher than the PVD threshold selected through the PLS[2:0] bits.
1: V\textsubscript{DD} is lower than the PVD threshold selected through the PLS[2:0] bits.

*Note: since the PVD is disabled in Standby mode, this bit is equal to 0 after Standby or reset until the PVDE bit is set.*

Bits 3:0  Reserved, must be kept at reset value.

### 6.8.3  **PWR control register 2 (PWR\_CR2)**

Address offset: 0x008

Reset value: 0x0000 0000

This register is not reset by wakeup from Standby mode, RESET signal and V\textsubscript{DD} POR. It is only reset by V\textsubscript{SW} POR and VSWRST reset.

This register must not be accessed when VSWRST bit in RCC\_BDCR register resets the V\textsubscript{SW} domain.

After reset, PWR\_CR2 register is write-protected. Prior to modifying its content, the DBP bit in PWR\_CR1 register must be set to disable the write protection.
6.8.4 **PWR control register 3 (PWR_CR3)**

Address offset: 0x00C  
Reset value: 0x0000 0006  
Reset by POR only, not reset by wakeup from Standby mode and RESET pad.

The lower byte of this register is written once after POR and must be written before changing VOS level or ck_sys clock frequency. No limitation applies to the upper bytes.

Programming data corresponding to an invalid combination of SMPSLEVEL, SMPSEXTHP SMPSEN, LDOEN and BYPASS bits (see Table 32) are ignored: data are not written, the written-once mechanism locks the register and any further write access is ignored. The default supply configuration is kept and the ACTVOSRDY bit in **PWR control status register 1 (PWR_CSR1)** goes on indicating invalid voltage levels. The system must be power cycled before writing a new value.

Illegal combinations of SMPSLEVEL, SMPSEXTHP, SMPSEN, LDOEN and BYPASS are described in Table 32.

The SMPS is not available on all packages. In this case, the SMPS is disabled.
### Power control (PWR)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>USB33RDY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>USBREGEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>USB33DEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Bit 26** **USB33RDY**: USB supply ready
- 0: USB33 supply not ready
- 1: USB33 supply ready

**Bit 25** **USBREGEN**: USB regulator enable
- 0: USB regulator disabled
- 1: USB regulator enabled

**Bit 24** **USB33DEN**: \( V_{DD33USB} \) voltage level detector enable
- 0: \( V_{DD33USB} \) voltage level detector disabled
- 1: \( V_{DD33USB} \) voltage level detector enabled

**Bits 23:17** Reserved, must be kept at reset value.

**Bit 16** **SMPSEXTRDY**: SMPS external supply ready
- This bit is set by hardware to indicate that the external supply from the SMPS is ready.
- 0: external supply not ready
- 1: external supply ready

**Bits 15:10** Reserved, must be kept at reset value.

**Bit 9** **VBRS**: \( V_{BAT} \) charging resistor selection
- 0: charge \( V_{BAT} \) through a 5 k\( \Omega \) resistor
- 1: charge \( V_{BAT} \) through a 1.5 k\( \Omega \) resistor

**Bit 8** **VBE**: \( V_{BAT} \) charging enable
- 0: \( V_{BAT} \) battery charging disabled
- 1: \( V_{BAT} \) battery charging enabled

**Bits 7:6** Reserved, must be kept at reset value.

**Bits 5:4** **SMPSLEVEL[1:0]**: SMPS voltage output level selection
- This bit is used when both the LDO and SMPS are enabled with SMPSEN and LDOEN enabled or when SMPSEXTHP is enabled. In this case SMPSLEVEL must be written with a value different than 00 at system startup.
- 00: reset value
- 01: 1.8 V
- 10: 2.5 V
- 11: 2.5 V
Power control (PWR) RM0455

6.8.5 PWR CPU control register (PWR_CPUCR)

This register allows controlling CPU domain power.

Address offset: 0x010
Reset value: 0x0000 0000

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

Bit 3 SMPSEXTHP: SMPS external power delivery selection
0: SMPS normal operating mode, no power delivery to external circuits
1: SMPS external operating mode, power delivery to external circuits

Bit 2 SMPSEN: SMPS enable
0: SMPS disabled
1: SMPS enabled (default)

Bit 1 LDOEN: low drop-out regulator enable
0: low drop-out regulator disabled
1: low drop-out regulator enabled (default)

Bit 0 BYPASS: power management unit bypass
0: power management unit normal operation
1: power management unit bypassed, voltage monitoring still active

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 RUN_SRD: prevent the SmartRun Domain (SRD) to enter Stop mode
0: SmartRun domain follows CPU subsystem modes.
1: SmartRun domain remains in Run mode regardless of CPU subsystem modes.

Bit 10 Reserved, must be kept at reset value.

Bit 9 CSSF: clear Standby and Stop flags (always read as 0)
This bit is cleared to 0 by hardware.
0: no effect
1: STOPF and SBF flags cleared

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 SBF: system Standby flag
This bit is set by hardware and cleared only by a POR or by setting the CSSF bit.
0: system has not been in Standby mode.
1: system has been in Standby mode.
Bit 5  **STOPF**: STOP flag
   This bit is set by hardware and cleared only by any reset or by setting the CSSF bit.
   0: system has not been in Stop mode.
   1: system has been in Stop mode.

Bits 4:3  Reserved, must be kept at reset value.

Bit 2  **PDDS_SRD**: system SmartRun domain power down Deepsleep
   This bit allows defining the Deepsleep mode for system SmartRun domain.
   0: Keeps Stop mode when SmartRun domain enters Deepsleep.
   1: Allows Standby mode when SmartRun domain enters Deepsleep.

Bit 1  Reserved, must be kept at reset value.

Bit 0  **RETDS_CD**: CPU domain power down Deepsleep selection.
   This bit defines the Deepsleep mode for CPU domain.
   0: Go to DStop mode when CPU domain enters Deepsleep.
   1: Go to DStop2 mode (deep retention) when CPU domain enters Deepsleep.

### 6.8.6  PWR SmartRun domain control register (PWR_SRDCR)

This register allows controlling SmartRun domain power.

Address offset: 0x018

Reset value: 0x0000 4000

Following reset, VOSRDY is read 1 by software.
Bits 31:16  Reserved, must be kept at reset value.

Bits 15:14  **VOS[1:0]:** voltage scaling selection according to performance
These bits control the V\textsubscript{CORE} voltage level and allow to obtain the best trade-off between power consumption and performance:
- In bypass mode, these bits must also be set according to the external provided core voltage level and related performance.
- When increasing the performance, the voltage scaling must be changed before increasing the system frequency.
- When decreasing performance, the system frequency must first be decreased before changing the voltage scaling.
00: scale 3 (default)
01: scale 2
10: scale 1
11: scale 0

Bit 13  **VOSRDY:** VOS ready bit for V\textsubscript{CORE} voltage scaling output selection
This bit is set to 1 by hardware when Bypass mode is selected in PWR\_CR3 register.
0: not ready, voltage level below VOS selected level
1: ready, voltage level at or above VOS selected level

Bits 12:0  Reserved, must be kept at reset value.

### 6.8.7  PWR wakeup clear register (PWR\_WKUPCR)

Address offset: 0x020
Reset value: 0x0000 0000
Reset only by system reset, not reset by wakeup from Standby mode.

Five wait states are required when writing this register. The AHB write access completes after the WKUPFx has been cleared.

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

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

<table>
<thead>
<tr>
<th>31:6</th>
<th>5:0</th>
<th>4:3</th>
<th>3:2</th>
<th>2:1</th>
<th>1:0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>31:6</th>
<th>5:0</th>
<th>4:3</th>
<th>3:2</th>
<th>2:1</th>
<th>1:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:6  Reserved, must be kept at reset value.

Bits 5:0  **WKUPCn+1:** clear wakeup pin flag for WKUPn+1
These bits are always read as 0.
0: no effect
1: writing 1 clears the WKUPFn+1 wakeup pin flag (bit is cleared to 0 by hardware).
### 6.8.8 PWR wakeup flag register (PWR_WKUPFR)

Address offset: 0x024  
Reset value: 0x0000 0000  
Reset only by system reset, not by wakeup from Standby mode.

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

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

Bits 31:6  Reserved, must be kept at reset value.  
Bits 5:0  **WKUPn+1**: wakeup pin WKUPn+1 flag  
This bit is set by hardware and cleared only by a RESET pin or by setting the WKUPCn+1 bit in PWR_WKUPCR register.  
0: no wakeup event occurred  
1: a wakeup event received from WKUPn+1 pin

### 6.8.9 PWR wakeup enable and polarity register (PWR_WKUPEPR)

Address offset: 0x028  
Reset value: 0x0000 0000  
Reset only by system reset, not by wakeup from Standby mode.

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

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

Bits 31:28  Reserved, must be kept at reset value.
Bits 27:16 **WKUPPUPD(truncate(n/2)-7)[1:0]**: wakeup pin pull configuration for WKUP(truncate(n/2)-7)

These bits define the I/O pad pull configuration used when WKUPEN(truncate(n/2)-7) = 1. The associated GPIO port pull configuration must be set to the same value or to 00.
The wakeup pin pull configuration is kept in Standby mode.
- 00: no pull-up
- 01: pull-up
- 10: pull-down
- 11: reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **WKUPnP-7**: wakeup pin polarity bit for WKUPn-7

These bits define the polarity used for event detection on WKUPn-7 external wakeup pin.
- 0: detection on high level (rising edge)
- 1: detection on low level (falling edge)

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **WKUPn+1**: enable wakeup pin WKUPn+1

These bit are set and cleared by software.
- 0: An event on WKUPn+1 pin does not wakeup the system from Standby mode.
- 1: A rising or falling edge on WKUPn+1 pin wakes up the system from Standby mode.

*Note: An additional wakeup event is detected if WKUPn+1 pin is enabled (by setting the WKUPn+1 bit) when WKUPn+1 pin level is already high when WKUPnP+1 selects rising edge, or low when WKUPnP+1 selects falling edge.*

### 6.8.10 PWR register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 31-0</th>
<th>Bits 31-20</th>
<th>Bits 19-8</th>
<th>Bits 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>PWR_CR1</td>
<td>SRAMSO</td>
<td>HSITF_SO</td>
<td>GFXSO</td>
<td>IT_CASSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AHBRA2SC</td>
<td>AHBRA1SC</td>
<td>AHBRA0SO</td>
<td>AHBRA0SO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AHBRA1SO</td>
<td>AHBRA2SO</td>
<td>AVDEN</td>
<td>AVD_READY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BOOSTE</td>
<td>DBP</td>
<td>PVDE</td>
<td>LVDS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMCVDO</td>
<td>AVDO</td>
<td>ACTVOSRDY</td>
<td>ACTVOS[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BRRDY</td>
<td>MONEN</td>
<td>LDOEN</td>
<td>BYPASS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USB33RDY</td>
<td>USB33DEN</td>
<td>VBRS</td>
<td>VBE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SMPSLEVEL[1:0]</td>
<td>SMPSEXTRDY</td>
<td>SMPSEXTHP</td>
<td>SIMPSEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAMPSEL[1:0]</td>
<td>VBE</td>
<td>VBE</td>
<td>LDOEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMODE</td>
<td>LVDS</td>
<td>LVDS</td>
<td>LVDS</td>
</tr>
</tbody>
</table>

**Reset value**: 0000000000001100 00000000

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 31-0</th>
<th>Bits 31-20</th>
<th>Bits 19-8</th>
<th>Bits 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x004</td>
<td>PWR_CSR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMCO</td>
<td>AHBRA2SC</td>
<td>AHBRA1SC</td>
<td>AHBRA0SO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AHBRA2SO</td>
<td>AHBRA1SO</td>
<td>AHBRA0SO</td>
<td>AVDEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BOOSTE</td>
<td>DBP</td>
<td>PVDE</td>
<td>LVDS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMCVDO</td>
<td>AVDO</td>
<td>ACTVOSRDY</td>
<td>ACTVOS[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BRRDY</td>
<td>MONEN</td>
<td>LDOEN</td>
<td>BYPASS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USB33RDY</td>
<td>USB33DEN</td>
<td>VBRS</td>
<td>VBE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SMPSLEVEL[1:0]</td>
<td>SMPSEXTRDY</td>
<td>SMPSEXTHP</td>
<td>SIMPSEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAMPSEL[1:0]</td>
<td>VBE</td>
<td>VBE</td>
<td>LDOEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMODE</td>
<td>LVDS</td>
<td>LVDS</td>
<td>LVDS</td>
</tr>
</tbody>
</table>

**Reset value**: 0000000000001000

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 31-0</th>
<th>Bits 31-20</th>
<th>Bits 19-8</th>
<th>Bits 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x008</td>
<td>PWR_CR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TBPY</td>
<td>TEMP</td>
<td>BRDY</td>
<td>BYPASS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Reset value**: 0000000000000000

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 31-0</th>
<th>Bits 31-20</th>
<th>Bits 19-8</th>
<th>Bits 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00C</td>
<td>PWR_CR3</td>
<td>USB3RDY</td>
<td>USB3DEN</td>
<td>VBRS</td>
<td>VBE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SMPSLEVEL[1:0]</td>
<td>SMPSEXTRDY</td>
<td>SMPSEXTHP</td>
<td>SIMPSEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAMPSEL[1:0]</td>
<td>VBE</td>
<td>VBE</td>
<td>LDOEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMODE</td>
<td>LVDS</td>
<td>LVDS</td>
<td>LVDS</td>
</tr>
</tbody>
</table>

**Reset value**: 0000000000001100
Table 47. Power control register map and reset values (continued)

| Offset | Register name       | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x010  | PWR_CPU CR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x014  | Reserved            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x018  | PWR_SRDCR           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x020  | PWR_WKUPCR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x024  | PWR_WKUPFR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x028  | PWR_WKUPEPR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.3 on page 128 for the register boundary addresses.
7 Low-power SRD domain

This section describes, through an example, how to use the SRD domain to implement low-power applications.

7.1 Introduction

The first part of the description explains how the EXTI, RCC and PWR blocks interact with each other and with the other system blocks. A detailed explanation on how the DMAMUX2 can be used to free the CPU is also provided.

The second part explains how to use the Autonomous mode to perform simple data transfers through an example of LPUART1 transmission.

Register programming is detailed only for the blocks related to the Autonomous mode.

7.2 EXTI, RCC and PWR interconnections

*Figure 38* shows the main EXTI, RCC and PWR interconnections.
Figure 38. EXTI, RCC and PWR interconnections
7.2.1 Interrupts and wakeup

Three kinds of signals are exchanged between the peripherals. They can be used to wake up the system from Stop mode:

- **Wakeup events** (or asynchronous interrupts)
  
  Some peripherals can generate interrupt events, even if their bus interface clock is not present. These interrupt events are called wakeup events (or asynchronous interrupts).
  
  Example: i2c1_wkup, usart1_wkup and lptim1_wkup.

- **Signals**
  
  Some peripherals generate a pulse instead of an interrupt signal. These pulses are called signals.
  
  Examples: lptim2_out and lptim3_out.

- **Interrupts**
  
  Contrary to signals, the interrupts should be cleared by a CPU or any other bus master, either by clearing the corresponding event bit in the peripheral register or by updating the FIFO interrupt level.

  All the interrupts associated to system peripherals are directly connected to the NVIC, except for the peripherals which are able to wake up the system from Stop mode or the CPU from CStop. In this latter case, the interrupts, signals or wakeup events are connected to the NVIC via the EXTI.
  
  Example: spi1_it, tim1_brk_it and tim1_upd_it.

The interrupt and wakeup sources that require to be cleared in the peripheral itself are connected to EXTI Direct Event inputs. The EXTI does not manage any CPU status pending bit.

The peripherals signals are connected to EXTI Configurable Event inputs. These EXTI inputs provide a CPU status pending bit which needs to be cleared by the application.

7.2.2 Block interactions

**Interaction between EXTI and PWR blocks**

The EXTI delivers wakeup requests signals (**exti_cd_wkup**, **exti_srd_wkup**) to the PWR controller. These signals are activated according to the state of the interrupts, signals or wakeup events connected to the EXTI. These wakeup requests are used by the PWR controller to supply the domain who needs to handle the activated wakeup event generated by the peripherals.

**Interaction between PWR and RCC blocks**

The PWR block controls the V_CORE supply according to the system operating mode (CRun, CSleep or CStop). The PWR block also controls the power switcher (ePODs) to set the CD domain in Retention mode (DStop2).

The RCC block controls the clock generation in accordance with the system operating mode. It is also responsible for reset generation.
To synchronize the system mode transitions, the RCC block is tightly coupled with the PWR controller:

- The RCC informs the PWR controller when peripherals located in Dx domain are allocated by the CPU (c_per_alloc_cd).
- The RCC also warns the PWR block when a domain clock is activated/deactivated. These signals are used in case of domain transition from DRun to DStop, DStop2 and Standby. In this case, the PWR controller waits until the domain clock has been gated, before switching down this domain.
- Similarly, the PWR controller informs the RCC about the VCore supply status of each domain (pwr_srd_wkup and pwr_cd_wkup). This information is used by the RCC when a domain transition from DStop or DStop2 to DRun occurs.

**Interaction between EXTI and SRD domain**

All the wakeup event inputs received by the EXTI from the peripherals located in SRD domain are forwarded back to the SRD domain after system clock re-synchronization. These events are used by the SRD domain to perform autonomous operations without activating the CPU.

The EXTI SRD_PenClear[3:0] inputs received from the SRD domain are used to acknowledge the ongoing wakeup requests generated by peripherals located in the SRD domain. The SRD_PenClear[3:0] inputs allow switching the system SRD domain from Run to Stop mode.

### 7.2.3 Role of SRD domain DMAMUX2

The DMAMUX2 implemented in the SRD domain allows chaining BDMA2 transfers. BDMA2 requests are synchronized thanks to trigger events (dmamux2_evtx) which can be generated when the expected amount of data has been transferred.

These events can also trigger DMAMUX2 request generators (REQ_GEN[3:0]), and thus chain several BDMA2 transfers. In fact REQ_GEN[3:0] can be triggered indirectly by all the wakeup events generated by all SRD domain peripherals.

Like LPTIM3 and LPTIM2 outputs, dmamux2_evt7 and dmamux2_evt6 events are connected to the EXTI. They can be used to switch the SRD domain from DRun to DStop or DStop2 mode when the task requested by the wakeup event is complete.
7.3 **Low-power application example based on LPUART1 transmission**

This section illustrates, through an example, the benefit of the SRD domain usage on power consumption. To help the user program the device, only the key register settings are given herein.

Refer to Sections *Reset and clock control (RCC)* and *Power control (PWR)* for additional details.

7.3.1 **Memory retention**

The SRD domain features 32 Kbytes of SRAM (SRD SRAM). When the CD domain is in DStop2 mode while the system is in Run mode, data can be transferred between this RAM area and to/from a peripheral located in the SRD domain.

In DStop2 mode, all the CD domain clocks are OFF but data are retained. The power consumption in DStop2 mode can be further optimized by choosing to shut off some SRAMs with the consequence to loose their content (see *Section 6: Power control (PWR)*).

*Note:* SRD SRAM remains available as long as the system is not in Standby mode.

If the system is in Standby mode, it is still possible to use the BKUP_SRAM. However, its size is limited to 4 Kbytes.

7.3.2 **Memory-to-peripheral transfer using LPUART1 interface**

**Example description**

*Figure 39* shows the proposed implementation. At a regular time interval given by LPTIM2, the CPU wakes up from CStop mode (which domain is in DStop2). When the CPU is in Run mode, it prepares the data to be transmitted via LPUART1, transfers them to SRD SRAM, and goes back to CStop. The SRD domain is configured to perform data transfers via LPUART1 and go back to Stop mode when the transfer is complete.

The LPTIM2 interface is used to wake up the system from Standby at regular time intervals.

The CPU must then perform the following operations:

1. Recover the application from the system Standby mode (RECO).
2. Process the new data to be sent via LPUART1 (PROC).
3. Transfer the data into SRD SRAM (XFER).
4. Configure the DMAMUX2, the BDMA2, the LPUART1, and the RCC (CFG).
5. Configure the EXTI (CFG).
6. Configure the PWR block to allow the CD domain to go to DStop2 mode.
7. Set the CPU to Stop mode.

The SRD domain executes the following tasks in Autonomous mode:

1. Transfer the data from SRD SRAM to LPUART1, using BDMA2.
2. When the LPUART1 interface indicates that the last byte has been transferred, the SRD domain is switched to Stop mode.
Figure 39. Timing diagram of SRD SRAM-to-LPUART1 transfer with BDMA2 and SRD domain in Autonomous mode

<table>
<thead>
<tr>
<th>Lptim2_wkup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU processing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CD State</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
</tr>
<tr>
<td>DRun</td>
</tr>
<tr>
<td>DStop2</td>
</tr>
<tr>
<td>DRun</td>
</tr>
<tr>
<td>DStop2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SRD processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM to LPUART1 Transfer</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>MEM to LPUART1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System / SRD State</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
</tr>
<tr>
<td>Run</td>
</tr>
<tr>
<td>Stop</td>
</tr>
<tr>
<td>Run</td>
</tr>
</tbody>
</table>

Note: The RUN_SRD bit can be used to force the SRD domain to remain in Run mode when the CD domain enters DStop or DStop2 mode. However it does not work in the above example since the SRD domain is not able to switch from Run to DStop2 mode on its own.

The proposed solution is explained in details in the following sections. It consists in setting a wakeup event for the SRD domain before the CD domain enters DStop2 mode. In this way the SRD domain remains in run mode and enters Stop mode only when this request is cleared.

### RCC programming

Before entering Autonomous mode (CD domain in DStop/DStop2 mode with SRD domain in Run mode), the RCC must be programmed to request the needed clocks to remain enabled. In this example, they are required for the SRD SRAM, BDMA2/DMAMUX2, LPTIM2, GPIOs and LPUART1

LPUART1 can use its own APB clock as kernel clock. Since the system will not enter Stop mode before LPUART1 has completed data transfer, PLLx can be used to provide clocks to the peripherals.

### PWR programming

In this example, the PWR block must be programmed in order to:

- Prevent system SRD domain to enter Standby mode when the data transfer is complete.
- Allow the CD domain to enter DStop2.
- Define the working voltage according to system modes.

Note: SRD domain could enter Standby as well, but in this case the LPTIM2 could not be used to wake up the system and the AWU should be used instead. In addition, everything must be reprogrammed when the system wakes up.
EXTI programming

The EXTI block must be configured to provide the following services:

- Keep SRD domain running when CD domain is in DStop/DStop2. This will be done by a software event.
- Set the device to Stop mode when the data transfer via LPUART1 is complete.
- Wake up the product from Stop when LPTIM2 time interval has elapsed.

The EXTI block is configured once before performing the first data transfer. For incoming data transfers, the programmed configuration remains unchanged; only some events need to be triggered or acknowledged.

Note: The CPU uses the event input number 0 to generate a software event. LPTIM2 wakeup signal is connected to event input number 48 (direct event input).

All other event inputs must be disabled: EXTI_RTSRx_TRy = ‘0’ and EXTI_FTSRx_TRy = ‘0’.

To generate a wakeup event for SRD domain, the CPU must write SWIER0 bit of EXTI_SWIER1 to ‘1’.

BDMA2 and DMAMUX2 programming

Two BDMA2 channels are required to execute data transfers via LPUART1.

- A BDMA2 channel, such as channel 0, is used to transfer data from SRD SRAM to LPUART1, using the TXE flag.
- The second BDMA2 channel role is to switch the SRD domain to Stop mode. For that purpose, DMAMUX2 request generator channel 0 (REQ_GEN0) and DMAMUX2 channel 7 synchronization block (SYNC7) are used in conjunction with BDMA2 channel 7.

BDMA2 channel 0 does not use DMAMUX2 trigger capabilities. Refer to Table 48 for initialization details.

BDMA2 channel 7 uses REQ_GEN0 to generate BDMA2 requests. The generation of BDMA2 requests is triggered by the LPUART1 transmit interrupt (lpuart1.tx_it). The LPUART1 interface generates lpuart1.tx_it interrupt when the transmit complete event is detected. The BDMA2 then clears the pending interrupt by performing a write operation to the LPUART1.

The SYNC7 block is programmed in Free-running mode. It generates a pulse on its dmamux2_evt7 output when the BDMA2 request generated by the REQ_GEN0 is complete. dmamux2_evt7 signal is used by the EXTI to switch back the SRD domain to Stop mode.

Figure 40 shows the active signal paths via DMAMUX2. The grayed blocks represent the unused paths.
Table 48 explain how to program BDMA2 and DMAMUX2 key functions. The way errors are handled is not described.

### Table 48. BDMA2 and DMAMUX2 initialization sequence (DMAMUX2\_INIT)

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Register content</th>
<th>Related actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAMUX2 SYNC0</td>
<td>DMAREQ_ID of DMAMUX2_C0CR = '10'</td>
<td>Selects LPUART_TX BDMA2 request.</td>
</tr>
<tr>
<td></td>
<td>SE of DMAMUX2_C0CR = '0'</td>
<td>Disables block synchronization.</td>
</tr>
<tr>
<td></td>
<td>EGE of DMAMUX2_C0CR = '0'</td>
<td>No event generation.</td>
</tr>
<tr>
<td></td>
<td>NBREQ of DMAMUX2_C0CR = '0'</td>
<td>Generates an event every BDMA2 transfer (Free-running mode).</td>
</tr>
<tr>
<td>DMAMUX2 SYNC7</td>
<td>DMAREQ_ID of DMAMUX2_C7CR = '0'</td>
<td>Selects of REQ_GEN0 as BDMA2 request.</td>
</tr>
<tr>
<td></td>
<td>SE of DMAMUX2_C7CR = '0'</td>
<td>Disables block synchronization.</td>
</tr>
<tr>
<td></td>
<td>EGE of DMAMUX2_C7CR = '1'</td>
<td>Enables event generation.</td>
</tr>
<tr>
<td></td>
<td>NBREQ of DMAMUX2_C7CR = '0'</td>
<td>Generates an event every BDMA2 transfer (Free-running mode).</td>
</tr>
<tr>
<td>DMAMUX2 REQ_GEN0</td>
<td>SIG_ID of DMAMUX2_RG0CR = '0d24'</td>
<td>Selects LPUART TX interrupt as trigger.</td>
</tr>
<tr>
<td></td>
<td>GPOL of DMAMUX2_RG0CR = '0b01'</td>
<td>Trigger on rising edge of the event.</td>
</tr>
<tr>
<td></td>
<td>GNBREQ of DMAMUX2_RG0CR = '0'</td>
<td>Generates only one BDMA2 request.</td>
</tr>
<tr>
<td></td>
<td>GE of DMAMUX2_RG0CR = '1'</td>
<td>Enables generator.</td>
</tr>
</tbody>
</table>
Table 48. BDMA2 and DMAMUX2 initialization sequence (DMAMUX2_INIT) (continued)

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Register content</th>
<th>Related actions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BDMA2 - CH0</strong></td>
<td>NDT bits of BDMA_CNDTR0 = DatNber PA of BDMA_CPAR0 = &amp;LPUART1_TDR MA of BDMA_CMAR0 = &amp;DatBuff DIR of BDMA_CCR0 = ‘1’ CIRC of BDMA_CCR0 = ‘0’ PINC of BDMA_CCR0 = ‘0’ MINC of BDMA_CCR0 = ‘1’ PSIZE of BDMA_CCR0 = ‘0’ MSIZE of BDMA_CCR0 = ‘1’ MEM2MEM of BDMA_CCR0 = ‘0’</td>
<td>Number of data to transfer. Address of LPUART1_TDR. Address of memory buffer of SRD SRAM. Read from memory. Circular mode disabled. Peripheral increment disabled. Memory increment enabled. Peripheral size = 8 bits. Memory size = 8 bits. Memory to memory disabled.</td>
</tr>
<tr>
<td><strong>BDMA2 - CH7</strong></td>
<td>NDT bits of BDMA_CNDTR7 = ‘1’ PA of BDMA_CPAR7 = &amp;LPUART1_ICR MA of BDMA_CMAR7 = &amp;DatClrTC DIR of BDMA_CCR7 = ‘1’ CIRC of BDMA_CCR7 = ‘0’ PINC of BDMA_CCR7 = ‘0’ MINC of BDMA_CCR7 = ‘1’ PSIZE of BDMA_CCR7 = 2 MSIZE of BDMA_CCR7 = 2 MEM2MEM of BDMA_CCR7 = ‘0’</td>
<td>Only one data transferred. Address of LPUART1_ICR (Interrupt Flag Clear Reg.). Address of a variable located into SRD SRAM. This variable must contain 0x0040 in order to clear the TC flag. Read from memory. Circular mode disabled. Peripheral increment disabled. Memory increment disabled. Peripheral size = 32 bits. Memory size = 32 bits. Memory to memory disabled.</td>
</tr>
</tbody>
</table>

**LPTIM2 programming**

When LPTIM2 wakeup event occurs, the CPU reboots and SRD domain mode is also set to Run mode.

An interrupt issued by LPTIM2 is pending on the CPU NVIC. LPTIM2 interrupt handler must acknowledge this LPTIM2 interrupt by writing ARRMCF bit in LPTIM2_ICR register to ‘1’ (LPTIM2_Ack).

**LPUART programming**

In the use-case described herein, the capability of the LPUART1 to request the kernel clock according to some events is not used.

LPUART1 is programmed so that is generates a BDMA request when its TX-FIFO is not full.

LPUART1 also generates an interrupt when the TX-FIFO and its transmit shift register are empty. This interrupt is used to switch the SRD domain to Stop mode.

Table 49 gives the key settings concerning the handling of Stop mode for LPUART1.
Respect the sequence described in Table 50 to enable LPUART1.

### Table 49. LPUART1 Initial programming (LPUART1_INIT)

<table>
<thead>
<tr>
<th>Register content</th>
<th>Related actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIFOEN of LPUART1_CR1 = '1'</td>
<td>Enables FIFO. BDMA2 will then use TXFNF (TXFIFO Not Full) flag for generating the BDMA2 requests.</td>
</tr>
<tr>
<td>TCIE of LPUART1_CR1 = '0'</td>
<td>Disables interrupt when the transmit buffer is empty.</td>
</tr>
<tr>
<td>UE of LPUART1_CR1 = '1'</td>
<td>Enables BDMA2.</td>
</tr>
<tr>
<td>TE of LPUART1_CR1 = '1'</td>
<td>Enables the LPUART1.</td>
</tr>
<tr>
<td>TXE of LPUART1_CR1 = '1'</td>
<td>Enables transmission.</td>
</tr>
<tr>
<td>DMAT of LPUART1_CR3 = '1'</td>
<td>Enables the BDMA2 mode for transmission.</td>
</tr>
</tbody>
</table>

### Table 50. LPUART1 start programming (LPUART1_Start)

<table>
<thead>
<tr>
<th>Register content</th>
<th>Related actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCCF of LPUART1_ICR = '1'</td>
<td>Clears the TC flag, to avoid immediate interrupt generation, which would clear the SRD_PendClear[1] in EXTI.</td>
</tr>
<tr>
<td>TCIE of LPUART1_CR1 = '1'</td>
<td>Enables interrupt when the transmit buffer is empty.</td>
</tr>
</tbody>
</table>

#### 7.3.3 Overall description of the low-power application example based on LPUART1 transmission

After a Power-on reset, the CPU perform the following operations:

1. Boot sequence (not described here).
2. Full initialization of RCC, PWR, EXTI, LPUART1, GPIOs, LPTIM2, DMAMUX2, BDMA2 and NVIC.
   - Only the relevant steps of RCC, EXTI, PWR, LPUART1, BDMA2 and DMAMUX2 initialization related to the Autonomous mode are described herein. Refer to the previous sections for additional details.
3. The CPU processes the data to be transferred and copies them to SRD SRAM.
4. The CPU generates a wakeup event (EXTI_Event) to maintain SRD in Run mode when CD enters DStop2.
5. The CPU enables the BDMA2 to start LPUART transmission and goes to CStop mode.
   - As it is allowed to do so, CD domain enters DStop2 while SRD remains in Run mode.
   - The data stored in SRD SRAM are retained while the CD domain is in DStop2 mode.
6. As soon as the BDMA2 is enabled, it serves the request from LPUART1 in order to fill its TX-FIFO. In parallel, serial data transmission can start.
7. When the expected amount of data has been transmitted (NDT bits of BDMA_CNDTR0 set to 0), the BDMA2 no longer provides data to the LPUART1. The LPUART1 generates an interrupt when the TX-FIFO and the transmit buffer are empty.
8. This interrupt triggers DMAMUX2_REQ_GEN0, thus activating a data transfer via BDMA2 channel 7 (BDMA_Ch7). This transfer clears LPUART1 TC flag, and the lputart1_tx_it is reset to '0'.
9. The end of this transfer triggers a dmamux2_evt7 signal which is used to clear the wakeup request generated by the CPU.
10. As a consequence, the SRD domain (i.e. the system) enters Stop mode and the system clock is gated. LPTIM2 still operates since it uses ck_lsi clock.
11. LPTIM2 lptim2_wkup interrupt wakes up the system. The device exits from Stop mode with the HSI clock. The CPU must restore the proper clock configuration during the warm re-boot sequence and perform the following tasks:
   a) Acknowledge LPTIM2 wakeup interrupt,
   b) Process the next data block and transfers them to SRD SRAM,
   c) Generate again a wakeup event for SRD domain,
   d) Start the BDMA2.
   e) Go back to CStop mode.

Note: The CPU does not need to initialize BDMA2, DMAMUX2 and LPUART1 again.

Figure 41. Timing diagram of LPUART1 transmission with SRD domain in Autonomous mode

7.3.4 Alternate implementations

More power efficient implementations are also possible. As an example the system clock can be stopped once the data have been transferred to LPUART1 TX-FIFO, instead of remaining activated during the whole transmission as in the example presented above. In this case, the LPUART1 must use ck_hsi or ck_csi as kernel clock when the system switches from Run to Stop mode. LPUART1 must be programmed to wake up SRD domain when its TX-FIFO in almost empty. This asynchronous interrupt can be used as trigger by the REQ_GENx of the DMAMUX2, which will perform a given number (e.g. 14) of data transfers to LPUART1 TDR and then switch back the SRD domain to Stop mode. This implementation is possible because the LPUART1 can request the kernel clock as long as the TX-FIFO and transmit buffer are not empty.
Further power consumption reduction during DStop/DStop2 mode can be achieved by shutting off the SRAMs located in the CD domain. Their content is then lost (see Section 6: Power control (PWR) for more details).

7.4 Other low-power applications

Other peripherals located in SRD domain, such as I2C4, SPI6/I2S6, DFSDM2 or DAC2, can be used to implement low-power applications.
8 Reset and clock control (RCC)

The RCC block manages the clock and reset generation for the whole microcontroller. The RCC block is located in the SmartRun domain (refer to Section 6: Power control (PWR) for a detailed description).

The operating modes this section refers to are defined in Section 6.6.1: Operating modes of the PWR block.

8.1 RCC main features

Reset block

- Generation of local and system reset
- Bidirectional pin reset allowing to reset the microcontroller or external devices
- WWDG and IWDG reset supported
- Power-on (POR) and Brownout (BOR) resets initiated by the power control (PWR)

Clock generation block

- Generation and dispatching of clocks for the complete device
- 3 separate PLLs using integer or fractional ratios
- Possibility to change the PLL fractional ratios on-the-fly
- Smart clock gating to reduce power dissipation
- 2 external oscillators:
  - High-speed external oscillator (HSE) supporting a wide range of crystals from 4 to 50 MHz frequency
  - Low-speed external oscillator (LSE) for the 32 kHz crystals
- 4 internal oscillators
  - High-speed internal oscillator (HSI)
  - 48 MHz RC oscillator (HSI48)
  - Low-power internal oscillator (CSI)
  - Low-speed internal oscillator (LSI)
- Buffered clock outputs for external devices
- Generation of two types of interrupts lines:
  - Dedicated interrupt lines for clock security management
  - One general interrupt line for other events
- Clock generation handling in Stop and Standby mode
- SmartRun domain autonomous mode
8.2 RCC block diagram

Figure 42 shows the RCC block diagram.

8.3 RCC pins and internal signals

Table 51 lists the RCC inputs and output signals connected to package pins or balls.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRST</td>
<td>I/O</td>
<td>System reset, can be used to provide reset to external devices</td>
</tr>
<tr>
<td>OSC32_IN</td>
<td>I</td>
<td>32 kHz oscillator input</td>
</tr>
<tr>
<td>OSC32_OUT</td>
<td>O</td>
<td>32 kHz oscillator output</td>
</tr>
<tr>
<td>OSC_IN</td>
<td>I</td>
<td>System oscillator input</td>
</tr>
</tbody>
</table>
The RCC exchanges a lot of internal signals with all components of the product, for that reason, Table 52 only shows the most significant internal signals.

### Table 51. RCC input/output signals connected to package pins or balls (continued)

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSC_OUT</td>
<td>O</td>
<td>System oscillator output</td>
</tr>
<tr>
<td>MCO1</td>
<td>O</td>
<td>Output clock 1 for external devices</td>
</tr>
<tr>
<td>MCO2</td>
<td>O</td>
<td>Output clock 2 for external devices</td>
</tr>
<tr>
<td>I2S_CKIN</td>
<td>I</td>
<td>External kernel clock input for digital audio interfaces: SPI/I2S, SAI, and DFSDM</td>
</tr>
<tr>
<td>USB_PHY1</td>
<td>I</td>
<td>USB clock input provided by the external USB_PHY</td>
</tr>
</tbody>
</table>

### Table 52. RCC internal input/output signals

<table>
<thead>
<tr>
<th>New signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rcc_it</td>
<td>O</td>
<td>General interrupt request line</td>
</tr>
<tr>
<td>rcc_hsecss_it</td>
<td>O</td>
<td>HSE clock security failure interrupt</td>
</tr>
<tr>
<td>rcc_lsecss_it</td>
<td>O</td>
<td>LSE clock security failure interrupt</td>
</tr>
<tr>
<td>rcc_ckfail_evt</td>
<td>O</td>
<td>Event indicating that a HSE clock security failure is detected. This signal is connected to TIMERS.</td>
</tr>
<tr>
<td>nreset</td>
<td>I/O</td>
<td>System reset</td>
</tr>
<tr>
<td>iwdg_out_rst</td>
<td>I</td>
<td>Reset line driven by the IWDG, indicating that a timeout occurred</td>
</tr>
<tr>
<td>wwdg_out_rst</td>
<td>I</td>
<td>Reset line driven by the WWDG, indicating that a timeout occurred</td>
</tr>
<tr>
<td>pwr_bor_rst</td>
<td>I</td>
<td>Brownout reset generated by the PWR block</td>
</tr>
<tr>
<td>pwr_por_rst</td>
<td>I</td>
<td>Power-on reset generated by the PWR block</td>
</tr>
<tr>
<td>pwr_vsw_rst</td>
<td>I</td>
<td>Power-on reset of the VSW domain generated by the PWR block</td>
</tr>
<tr>
<td>rcc_perx_rst</td>
<td>O</td>
<td>Reset generated by the RCC for the peripherals</td>
</tr>
<tr>
<td>pwr_cd_wkup</td>
<td>I</td>
<td>Wakeup domain request generated by the PWR and used to restore the domain clocks</td>
</tr>
<tr>
<td>pwr_srd_wkup</td>
<td>I</td>
<td>Wakeup domain request generated by the PWR and used to restore the domain clocks</td>
</tr>
<tr>
<td>rcc_pwd_cd_req</td>
<td>O</td>
<td>Low-power request generated by the RCC and used to set the domain into low-power mode</td>
</tr>
<tr>
<td>rcc_pwd_srd_req</td>
<td>O</td>
<td>Low-power request generated by the RCC and used to set the domain into low-power mode</td>
</tr>
<tr>
<td>cpu_sleep</td>
<td>I</td>
<td>Signals generated by the CPU, indicating if the CPU is in CRun, CSleep or CStop</td>
</tr>
<tr>
<td>cpu_deepsleep</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>perx_ker_ckreq</td>
<td>I</td>
<td>Signal generated by some peripherals in order to request the activation of their kernel clock</td>
</tr>
<tr>
<td>rcc_perx_ker_ck</td>
<td>O</td>
<td>Kernel clock signals generated by the RCC, for some peripherals</td>
</tr>
</tbody>
</table>
8.4 RCC reset block functional description

The following sources can generate a reset:
- an external device via NRST pin
- a failure on the supply voltage applied to VDD
- a watchdog timeout
- a software command

The reset scope depends on the source that generates the reset.

Three reset categories exist:
- power-on/off reset
- system reset
- local resets

8.4.1 Power-on/off reset

The power-on/off reset (pwr_por_rst) is generated by the power controller block (PWR). It is activated when the input voltage (VDD) is below a threshold level. This is the most complete reset since it resets the whole circuit, except the Backup domain.

The power-on/off reset function can be disabled through PDR_ON pin (see Section 6.5: Power supply supervision).

Refer to Table 53: Reset distribution summary for details.

<table>
<thead>
<tr>
<th>New signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rcc_perx_bus_ck</td>
<td>O</td>
<td>Bus interface clock signals generated by the RCC for peripherals</td>
</tr>
<tr>
<td>rcc_bus_ck</td>
<td>O</td>
<td>Clocks for APB, AHB and AXI bridges generated by the RCC</td>
</tr>
<tr>
<td>rcc_cpu_ck</td>
<td>O</td>
<td>Clocks for the CPU, generated by the RCC</td>
</tr>
<tr>
<td>rcc_fclk_c</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>
8.4.2 System reset

A system reset (nreset) resets all registers to their default values unless otherwise specified in the register description.

A system reset can be generated from one of the following sources:

- a reset from NRST pin (external reset)
- a reset from the power-on/off reset block (pwr_por_rst)
- a reset from the brownout reset block (pwr_bor_rst)
  Refer to Section 6.5.2: Brownout reset (BOR) for a detailed description of the BOR function.
- a reset from the independent watchdogs (iwdg_out_rst)
- a software reset from the Cortex®-M7 core
  It is generated via the SYSRESETREQ signal issued by the Cortex®-M7 core. This signal is also named SFTRESET in this document.
- a reset from the window watchdogs depending on WWDG configuration (wwdg_out_rst)
- A reset from the low-power mode security reset, depending on option byte configuration (lpwr_rst)

Note: The SYSRESETREQ bit in Cortex®-M7 through the FPU application interrupt and reset control register, must be set to force a software reset on the device. Refer to the Cortex®-M7 with FPU technical reference manual for more details (see http://infocenter.arm.com).

As shown in Figure 43, some internal sources (such as pwr_por_rst, pwr_bor_rst, iwdg_out_rst) perform a system reset of the circuit, which is also propagated to the NRST pin to reset the connected external devices. The pulse generator guarantees a minimum reset pulse duration of 20 μs for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

Note: It is not recommended to let the NRST pin unconnected. When it is not used, connect this pin to ground via a 10 to 100 nF capacitor (C_R in Figure 43).

Figure 43. System reset circuit
8.4.3 Local resets

Domain reset

When the system exits from Standby mode, a \textit{stby\_rst} reset is applied. The \textit{stby\_rst} signal generates a reset of the complete V\textsubscript{CORE} domain as long the V\textsubscript{CORE} voltage provided by the internal regulator is not valid.

\textit{Table 53} gives a detailed overview of reset sources and scopes.

Table 53. Reset distribution summary

<table>
<thead>
<tr>
<th>Reset source</th>
<th>Reset name</th>
<th>CPU</th>
<th>CD interconnect</th>
<th>CD peripherals</th>
<th>CD debug</th>
<th>CD WWGD</th>
<th>SRD peripherals</th>
<th>CD IWDG</th>
<th>FLASH</th>
<th>RTC domain</th>
<th>Backup RAM</th>
<th>System supply</th>
<th>SRST pin</th>
<th>Comments</th>
</tr>
</thead>
</table>
| Pin          | NRST       | x x x - | x x x - | - - - - | x | | | | | | | | | -- Resets CPU domain and all its peripherals
| | | | | | | | | | | | | | | -- Resets SmartRun domain peripherals
| | | | | | | | | | | | | | | -- Resets V\textsubscript{DD} domain: IWDG, LDO
| | | | | | | | | | | | | | | -- Debug features, RTC and backup RAM are not reset.
| PWR          | pwr\_bor\_rst | x x x - | x x x - | - - - - | x | | | | | | | | | -- Same as pin reset. The pin is asserted as well.
| | | | | | | | | | | | | | | -- Same as \textit{pwr\_bor\_rst} reset, plus:
| | | | | | | | | | | | | | | Reset of the Flash memory digital block (including the option byte loading).
| | | | | | | | | | | | | | | Reset of the debug block
| | lpwr\_rst | x x x - | x x x - | - - - - | x | | | | | | | | | -- The low-power mode security reset has the same scope than \textit{pwr\_por\_rst}. Refer to \textit{Section 8.4.5: Low-power mode security reset (lpwr\_rst)} for additional information.
| RCC          | VSWRST     | - - - - | - - - - | - | - | - - | - - | - - | - - | - - | - - | - - | -- The Backup domain reset can be triggered by software. Refer to \textit{Section 8.4.6: Backup domain reset} for additional information.
| | stby\_rst | x x x x x - | - - - - | - - - - | - - - - | - - - - | - - - - | - - - - | - - - - | - - - - | - - - - | - - - - | -- When the device exits Standby mode, a reset of the complete V\textsubscript{CORE} domain is performed as long the V\textsubscript{CORE} voltage is not valid. The V\textsubscript{CORE} is supplied by the internal regulator.
| | | | | | | | | | | | | | | NRST signal is not asserted.
| CPU          | SFTRESET   | x x x - | x x x - | - - - - | x | | | | | | | | | -- This reset is generated by software when writing SYSRESETREQ bit located into AIRCR register of the Cortex\textsuperscript{\textregistered}\textendash M7 core.
| | | | | | | | | | | | | | | -- Same scope as \textit{pwr\_bor\_rst} reset
| Backup domain | pwr\_vsw\_rst | - - - - | - - - - | - - - - | x | | | | | | | | | -- This reset is generated by the Backup domain when the V\textsubscript{SW} supply voltage is outside the operating range.
| | iwdg\_out\_rst | x x x - | x x x - | - - - - | x | | | | | | | | | -- Same as \textit{pwr\_bor\_rst} reset
| | wwdg\_out\_rst | x x x - | x x x - | - - - - | x | | | | | | | | | -- Same as \textit{pwr\_bor\_rst} reset

8.4.4 Reset source identification

The CPU can identify the reset source by checking the reset flags in the RCC\_RSR register.
The CPU can reset the flags by setting RMVF bit.

*Table 54* shows how the status bits of the RCC_RSR register behave according to the situation that generated the reset. For example, when an IWDG timeout occurs (line #6), if the CPU is reading the RCC_RSR register during the boot phase, both PINRSTF and IWDGRSTF bits are set, indicating that the IWDG also generated a pin reset.

### Table 54. Reset source identification (RCC_RSR)$^{(1)}$

<table>
<thead>
<tr>
<th>#</th>
<th>Situations generating a reset</th>
<th>LPWRSTF</th>
<th>WWGRSTF</th>
<th>IWDGRSTF</th>
<th>SFTRSTF</th>
<th>PORRSTF</th>
<th>PINRSTF</th>
<th>BORRSTF</th>
<th>CDRSTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Power-on reset (pwr_por_rst)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Pin reset (NRST)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Brownout reset (pwr_bor_rst)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>System reset generated by CPU (SFTRESET)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>WWGDG reset (wwdg_out_rst)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>IWDG reset (iwdg_out_rst)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>CPU domain exits DStop mode and pin reset during DStop</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>CPU domain exits DStop2 mode and pin reset during DStop2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>CPU erroneously enters CStop mode</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

1. Grayed cells highlight the register bits that are set.

### 8.4.5 Low-power mode security reset (lpwr_rst)

To prevent critical applications from mistakenly enter a low-power mode, two low-power mode security resets are available. When enabled through nRST_STOP_CD and nRST_STANDBY option bytes, a system reset is generated if the following conditions are met:

- The CPU domain (CD) accidentally enters Dstop or Dstop2 mode.
  This type of reset is enabled by resetting nRST_STOP_CD user option byte. In this case, whenever the Dstop or Dstop2 mode entry sequence is successfully executed, a system reset is generated.
- CPU and system accidentally enter Standby mode.
  This type of reset is enabled by resetting nRST_Standby user option byte. In this case, whenever a Standby mode entry sequence is successfully executed, a system reset is generated.
  When the Standby mode is entered, a flag is also set in the power controller.

LPWRRSTF bit in the **RCC reset status register (RCC_RSR)** indicates that a low-power mode security reset occurred (see line #9 in *Table 54*).

**lpwr_rst** is activated when a low-power mode security reset due to CPU occurred.

Refer to *Section 4.4: FLASH option bytes* for additional information.

Refer to *Section 6: Power control (PWR)* for additional information and *Table 33: Low-power mode summary* for the overview of the existing power modes.
8.4.6 Backup domain reset

A Backup domain reset is generated when one of the following events occurs:

- A software reset, triggered by setting VSWRST bit in the **RCC Backup domain control register (RCC_BDCR)**. All RTC registers and the RCC_BDCR register are reset to their default values. The backup RAM is not affected.
- $V_{SW}$ voltage is outside the operating range. All RTC registers and the RCC_BDCR register are reset to their default values. In this case the content of the backup RAM is no longer valid.

There are two ways to reset the backup RAM:

- through the Flash memory interface by requesting a protection level change from 1 to 0
- when a tamper event occurs

Refer to **Section 6.4.4: Backup domain** section of PWR block for additional information.

8.4.7 Power-on and wakeup sequences

For detailed diagrams refer to **Section 6.4.1: System supply startup** in the PWR section.

The time interval between the event that exits the product from a low-power and the moment where the CPU is able to execute code, depends on the system state and on its configuration. **Figure 44** shows the most usual examples.

**Power-on wakeup sequence**

The power-on wakeup sequence shown in **Figure 44** gives the most significant phases of the power-on sequence. It is the longest sequence since the circuit was not powered. Note that this sequence remains unchanged, whatever $V_{BAT}$ was present or not.

**Boot from pin reset**

When a pin reset occurs, $V_{DD}$ is still present. As a result:

- The regulator settling time is faster since the reference voltage is already stable.
- The HSI restart delay may be needed if the HSI was not enabled when the NRST occurred, otherwise this restart delay phase is skipped.
- The Flash memory power recovery delay can also be skipped if the Flash memory was enabled when the NRST occurred.

**Note:** The boot sequence is similar for pwr_bor_rst, lpwr_rst, STFxRESET, iwdg_out_rst and wwdg_out_rst.

**Boot from system Standby**

When waking up from system Standby, the reference voltage is stable since $V_{DD}$ has not been removed. As a result, the regulator settling time is fast. Since $V_{CORE}$ was not present, the restart delay for the HSI, the Flash memory power recovery and the option byte reloading cannot be skipped.
Restart from system Stop

When restarting from system Stop (CPU domain can be in DStop or DStop2), V\textsubscript{DD} is still present. As a result, the sequence is mainly composed of two steps:

1. regulator settling time to reach VOS3 (default voltage)
2. HSI/CSI restart delay. This step can be skipped if HSIKERON or CSIKERON bit is set to 1 in the \textit{RCC source control register (RCC\_CR)}.

Restart from system Run with CPU domain in DStop/DStop2

Since the system is in run mode, the regulator is already activated.

The restart sequence of the CPU domain in DStop or DStop2 mode is mainly composed of two steps:

1. Handshake between RCC, EXTI and PWR blocks.
2. The power-switch (ePOD) settling time to change from retention mode to normal mode in case of DStop2.
### Restart from domain DStop

The restart sequence of a domain from DStop is mainly composed of the handshake between the RCC, EXTI and PWR blocks.

#### Figure 44. Boot sequences versus system states

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VDD &gt; POR</strong></td>
<td>Power-on wake-up (with or without VBAT)</td>
</tr>
<tr>
<td><strong>NRST goes HIGH</strong></td>
<td>PAD Reset (or pwr_bor_rst, SFTRESET, lwdg_out_rst, ...)</td>
</tr>
<tr>
<td><strong>Wakeup event</strong></td>
<td>Wake-up from system Standby</td>
</tr>
<tr>
<td><strong>Wakeup event</strong></td>
<td>Wake-up from system Stop</td>
</tr>
<tr>
<td><strong>Wakeup event</strong></td>
<td>Re-start from system Run with CPU Domain in DStop2</td>
</tr>
<tr>
<td><strong>Wakeup event</strong></td>
<td>Re-start from system Run with CPU Domain in DStop</td>
</tr>
</tbody>
</table>

**Notes:**
- REG_VOS3, HSI/CSI, RUN
- ePOD
- RUN
- FL_PWR, FL_OPTB
- Bandgap and regulator settling time
- REG setting time to reach the VOS3
- Flash power recovery delay
- Option bytes loading delay
- Power switch delay
- CPU Fetch
- Delay due to handshake RCC, PWR and AIEC
- HSI or CSI Restart delay
8.5 RCC clock block functional description

The RCC provides a wide choice of clock generators:

- HSI (high-speed internal oscillator) clock: ~ 8, 16, 32 or 64 MHz
- HSE (high-speed external oscillator) clock: 4 to 48 MHz
- LSE (low-speed external oscillator) clock: 32 kHz
- LSI (low-speed internal oscillator) clock: ~ 32 kHz
- CSI (low-power internal oscillator) clock: ~4 MHz
- HSI48 (high-speed internal oscillator) clock: ~48 MHz

The RCC offers then a high flexibility for the application to select the appropriate clock for CPU and peripherals, in particular for peripherals that require a specific clock such as USB OTG-FS and HS, SPI/I2S, SAI and SDMMC.

To optimize the power consumption, each clock source can be switched ON or OFF independently.

The RCC provides up to 3 PLLs; each of them can be configured with integer or fractional ratios.

As shown in the Figure 45, the RCC offers two clock outputs (MCO1 and MCO2), with a great flexibility on the clock selection and frequency adjustment.

The SCGU block (system clock generation unit) contains several prescalers used to configure the CPU and bus matrix clock frequencies.

The PKSU block (peripheral kernel clock selection unit) provides several dynamic switches allowing a large choice of kernel clock distribution to peripherals.

The PKEU (peripheral kernel clock enable unit) and SCEU (system clock enable unit) blocks perform the peripheral kernel clock gating, and the bus interface/cores/bus matrix clock gating, respectively.
Figure 45. Top-level clock tree

The selected input can be changed on-the-fly without spurs on the output signal. * Represents the selected mux input after a system reset.
8.5.1 Clock naming convention

The RCC provides clocks to the complete circuit. To avoid misunderstanding, the following terms are used in this document:

- **Peripheral clocks**
  The peripheral clocks are the clocks provided by the RCC to the peripherals. Two kinds of clock are available:
  - bus interface clocks
  - kernel clocks
  A peripheral receives from the RCC a bus interface clock in order to access its registers, and thus control the peripheral operation. This clock is generally the AHB, APB or AXI clock depending on which bus the peripheral is connected to. Some peripherals only need a bus interface clock (such as RNG, TIMx).
  Some peripherals also require a dedicated clock to handle the interface function. This clock is named kernel clock. As an example, peripherals such as SAI must generate specific and accurate master clock frequencies, which require dedicated kernel clock frequencies. Another advantage of decoupling the bus interface clock from the specific interface needs, is that the bus clock can be changed without reprogramming the peripheral.

- **CPU clock**
  The CPU clock is the clock provided to the CPU. It is derived from the system clock (sys_ck).

- **Bus matrix clocks**
  The bus matrix clocks are the clocks provided to the different bridges (APB, AHB or AXI). These clocks are derived from the system clock (sys_ck).

8.5.2 Oscillators description

**HSE oscillator**

The HSE block can generate a clock from two possible sources:

- external crystal/ceramic resonator
- external clock source

![Figure 46. HSE/LSE clock source](image-url)
External clock source (HSE bypass)

In this mode, an external clock source must be provided to OSC_IN pin. The external clock can be low swing (analog) or digital.

If this clock is directly used by a peripheral, the duty cycle requirement is defined by this peripheral and the application (refer to datasheet for more details).

This external clock is provided to the OSC_IN pin while the OSC_OUT pin must be left HI-Z (see Figure 46).

In case of an analog clock (low swing) the HSEBYP and HSEON bits must be set to 1 in the RCC source control register (RCC_CR).

In case of a digital clock, the HSEBYP and the HSEEXT bits must be set to 1 followed by setting the HSEON bit to 1 in the RCC source control register (RCC_CR).

External crystal/ceramic resonator

The oscillator is enabled by setting the HSEBYP bit to 0 and HSEON bit to 1.

The HSE can be used when the product requires a very accurate high-speed clock.

The associated hardware configuration is shown in Figure 46: the resonator and the load capacitors must be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The HSERDY flag of the RCC source control register (RCC_CR) indicates whether the HSE oscillator is stable or not. At startup, the hse_ck clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the RCC clock source interrupt enable register (RCC_CIER).
The HSE can be switched ON and OFF through the HSEON bit. Note that the HSE cannot be switched OFF if one of the two conditions is met:

- The HSE is used directly (via software mux) as system clock.
- The HSE is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the HSEON bit to 0.

The HSE is automatically disabled by hardware, when the system enters Stop or Standby mode (refer to Section 8.5.7: Handling clock generators in Stop and Standby modes for additional information).

In addition, the HSE clock can be driven to the MCO1 and MCO2 outputs and used as clock source for other application components.

**LSE oscillator**

The LSE block can generate a clock from two possible sources:

- external crystal/ceramic resonator
- external user clock

**External clock source (LSE bypass)**

In this mode, an external clock source must be provided to OSC32_IN pin. The input clock can have a frequency up to 1 MHz and be low swing (analog) or digital. A duty cycle close to 50% is recommended.

This external clock is provided to the OSC32_IN pin while the OSC32_OUT pin must be left HI-Z (see Figure 46).

In case of an analog clock (low swing) the LSEBYP and HSEON bits must be set to 1 (RCC Backup domain control register (RCC_BDCR)).

In case of a digital clock the LSEBYP and the LSEEXT bits must be set to 1 followed by setting the LSEON bit to 1 (RCC Backup domain control register (RCC_BDCR))

If the RTC is used, the LSE bypass must not be configured in digital mode but in low swing analog mode (default value after reset)

**External crystal/ceramic resonator (LSE crystal)**

The LSE clock is generated from a 32.768 kHz crystal or ceramic resonator. It has the advantage to provide a low-power highly accurate clock source to the real-time clock (RTC) for clock/calendar or other timing functions.

The LSERDY flag of the RCC Backup domain control register (RCC_BDCR) indicates whether the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the RCC clock source interrupt enable register (RCC_CIER).

The LSE oscillator is switched ON and OFF using the LSEON bit. The LSE remains enabled when the system enters Stop or Standby mode.

In addition, the LSE clock can be driven to the MCO1 output and used as clock source for other application components.

The LSE also offers a programmable driving capability (LSED RV[1:0]) that can be used to modulate the amplifier driving capability. This driving capability is chosen according to the external crystal/ceramic component requirement to insure a stable oscillation.
The driving capability must be set before enabling the LSE oscillator.

**HSI oscillator**

The HSI block provides the default clock to the product.

The HSI is a high-speed internal RC oscillator that can be used directly as system clock, peripheral clock, or as PLL input. A predivider allows the application to select an HSI output frequency of 8, 16, 32 or 64 MHz. This predivider is controlled by the HSIDIV.

The HSI advantages are the following:
- low-cost clock source since no external crystal is required
- faster startup time than HSE (a few microseconds)

The HSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

The HSI can be switched ON and OFF using the HSION bit. Note that the HSI cannot be switched OFF if one of the two conditions is met:
- The HSI is used directly (via software mux) as system clock.
- The HSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the HSION bit to 0.

Note that the HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to 1). In that case the hardware does not update the HSIDIV with the new value. However it is possible to change the HSIDIV if the HSI is used directly as system clock.

The HSIRDY flag indicates if the HSI is stable or not. At startup, the HSI output clock is not released until this bit is set by hardware.

The HSI clock can also be used as a backup source (auxiliary clock) if the HSE fails (refer to **CSS on HSE on page 348**). The HSI can be disabled or not when the system enters Stop mode. Refer to **Section 8.5.7: Handling clock generators in Stop and Standby modes** for additional information.

In addition, the HSI clock can be driven to the MCO1 output and used as clock source for other application components.

Care must be taken when the HSI is used as kernel clock for communication peripherals, the application must take into account the following parameters:
- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available,
- the frequency accuracy.

**Note:** The HSI can remain enabled when the system is in Stop mode (see **Section 8.5.7: Handling clock generators in Stop and Standby modes** for additional information).

HSION, HSIRDY and HSIDIV bits are located in the RCC source control register (**RCC_CR**).

**HSI calibration**

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. That is why each device is factory calibrated by STMicroelectronics to achieve an accuracy of ACC_HSI (refer to the product datasheet for more information).
After a power-on reset, the factory calibration value is loaded in the HSICAL[11:0] bits.

If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the HSI frequency using the HSITRIM[6:0] bits.

**Note:** HSICAL[11:0] and HSITRIM[6:0] bits are located in the RCC CSI calibration register (RCC_CSICFGR).

**Figure 48. HSI calibration flow**

**CSI oscillator**

The CSI is a low-power RC oscillator that can be used directly as system clock, peripheral clock, or PLL input.

The CSI advantages are the following:
- low-cost clock source since no external crystal is required
- faster startup time than HSE (a few microseconds)
- very low-power consumption,

The CSI provides a clock frequency of about 4 MHz, while the HSI is able to provide a clock up to 64 MHz.

CSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

The CSI can be switched ON and OFF through the CSION bit. The CSIRDY flag indicates whether the CSI is stable or not. At startup, the CSI output clock is not released until this bit is set by hardware.

The CSI cannot be switched OFF if one of the two conditions is met:
- The CSI is used directly (via software mux) as system clock.
- The CSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the CSION bit to 0.

The CSI can be disabled or not when the system enters Stop mode (refer to Section 8.5.7: Handling clock generators in Stop and Standby modes for additional information).

In addition, the CSI clock can be driven to the MCO2 output and used as clock source for other application components.
Even if the CSI settling time is faster than the HSI, care must be taken when the CSI is used as kernel clock for communication peripherals: the application must take into account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available,
- the frequency precision.

**Note:** CSION and CSIRDY bits are located in the RCC source control register (RCC_CR).

**CSI calibration**

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by STMicroelectronics to achieve accuracy of ACC\(_{CSI}\) (refer to the product datasheet for more information).

After reset, the factory calibration value is loaded in the CSICAL[7:0] bits.

If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the CSI frequency using the CSITRIM[5:0] bits.

**Note:** Bits CSICAL[7:0] and CSITRIM[5:0] are located into the RCC CSI calibration register (RCC_CSICFGR)

![Figure 49. CSI calibration flow](MSv48152V1)

**HSI48 oscillator**

The HSI48 is an RC oscillator delivering a 48 MHz clock that can be used directly as kernel clock for some peripherals.

The HSI48 oscillator mainly aims at providing a high-precision clock to the USB peripheral by means of a special clock recovery system (CRS) circuitry, that can use the USB SOF signal, the LSE or an external signal to automatically adjust the oscillator frequency on-the-fly, with a very small granularity.

The HSI48 oscillator is disabled as soon as the system enters Stop or Standby mode. When the CRS is not used, this oscillator is free running and thus subject to manufacturing process variations. That is why each device is factory calibrated by STMicroelectronics to achieve an accuracy of ACC\(_{HSI48}\) (refer to the product datasheet for more information).

For more details on how to configure and use the CRS, refer to **Section 9: Clock recovery system (CRS)**.

The HSI48RDY flag indicates whether the HSI48 oscillator is stable or not. At startup, the HSI48 output clock is not released until this bit is set by hardware.
The HSI48 can be switched ON and OFF using the HSI48ON bit.

The HSI48 clock can also be driven to the MCO1 multiplexer and used as clock source for other application components.

Note: HSI48ON and HSI48RDY bits are located in the RCC source control register (RCC_CR).

**LSI oscillator**

The LSI acts as a low-power clock source that can be kept running when the system is in Stop or Standby mode for the independent watchdog (IWDG) and auto-wakeup unit (AWU). The clock frequency is around 32 kHz. For more details, refer to the electrical characteristics section of the datasheet.

The LSI can be switched ON and OFF using the LSION bit. The LSIRDY flag indicates whether the LSI oscillator is stable or not. If an independent watchdog is started either by hardware or software, the LSI is forced ON and cannot be disabled.

The LSI remains enabled when the system enters Stop or Standby mode (refer to Section 8.5.7: Handling clock generators in Stop and Standby modes for additional information).

At LSI startup, the clock is not provided until the hardware sets the LSIRDY bit. An interrupt can be generated if enabled in the RCC clock source interrupt enable register (RCC_CIER).

In addition, the LSI clock can be driven to the MCO2 output and used as a clock source for other application components.

Note: Bits LSION and LSIRDY bits are located into the RCC clock control and status register (RCC_CSR).

### 8.5.3 Clock security system (CSS)

**CSS on HSE**

The clock security system can be enabled by software via the HSECSSON bit. The HSECSSON bit can be enabled even when the HSEON is set to 0.

The CSS on HSE is enabled by the hardware when the HSE is enabled and ready, and HSECSSON set to 1.

The CSS on HSE is disabled when the HSE is disabled. As a result, this function does not work when the system is in Stop mode.

It is not possible to clear directly the HSECSSON bit by software.

The HSECSSON bit is cleared by hardware when a system reset occurs or when the system enters Standby mode (see Section 8.4.2: System reset).

If a failure is detected on the HSE clock, the system automatically switches to the HSI in order to provide a safe clock. The HSE is then automatically disabled, a clock failure event is sent to the break inputs of advanced-control timers (TIM1, TIM8, TIM15, TIM16, and TIM17), and an interrupt is generated to inform the software about the failure (CSS interrupt: rcc_hsecss_it), thus allowing the MCU to perform rescue operations. If the HSE output was used as clock source for PLLs when the failure occurred, the PLLs are also disabled.

If an HSE clock failure occurs when the CSS is enabled, the CSS generates an interrupt that causes the automatic generation of an NMI. The HSECSSF flag in RCC clock source interrupt flag register (RCC_CIFR) is set to 1 to allow the application to identify the failure.
source. The NMI routine is executed indefinitely until the HSECSSF bit is cleared. As a consequence, the application must clear the HSECSSF flag in the NMI ISR by setting the HSECSSC bit in the **RCC clock source interrupt clear register (RCC_CICR)**.

**CSS on LSE**

A clock security system on the LSE oscillator can be enabled by software by programming the LSECSSON bit in the **RCC Backup domain control register (RCC_BDCR)**.

This bit can be disabled only by hardware one of the following conditions is met:
- after a pwr_vsw_rst (VSW software reset)
- after a failure detection on LSE.

The LSECSSON bit must be written after the LSE is enabled (LSEON bit set by software) and ready (LSERDY set by hardware), and after the RTC clock has been selected through the RTCSEL bit.

The CSS on LSE works in all modes (Run, Stop and Standby) except VBAT.

If an LSE failure is detected, the LSE clock is no more delivered to the RTC but the values of RTCSEL, LSECSSON and LSEON bits are not changed by the hardware.

A wakeup is generated in Standby mode. In other modes an interrupt (**rcc_lsecss_it**) can be sent to wake up the software. The software must then disable the LSECSSON bit, stop the defective LSE (clear LSEON bit) and can change the RTC clock source (no clock or LSI or HSE) through RTCSEL bits, or take any required action to secure the application.

### 8.5.4 Clock output generation (MCO1/MCO2)

Two microcontroller clock output pins (MCO), MCO1 and MCO2, are available. A clock source can be selected for each output. The selected clock can be divided thanks to configurable prescaler (refer to **Figure 45** for additional information on signal selection).

MCO1 and MCO2 outputs are controlled via MCO1PRE[3:0], MCO1[2:0], MCO2PRE[3:0] and MCO2[2:0] located in the **RCC clock configuration register (RCC_CFGR)**.

The GPIO port corresponding to each MCO pin must be programmed in alternate function mode.

The clock provided to the MCOs outputs must not exceed the maximum pin speed (refer to the product datasheet for information on the supported pin speed).

### 8.5.5 PLL description

The RCC features three PLLs:
- a main PLL, PLL1, generally used to provide clocks to the CPU and some peripherals
- two dedicated PLLs, PLL2 and PLL3, used to generate the kernel clock for peripherals
The PLLs integrated into the RCC are completely independent. They offer the following features:

- A VCO supporting two modes:
  - A wide-range
  - A low-range used for instance in audio application cases
- Input frequency range:
  - 2 to 16 MHz for the VCO in wide-range mode
  - 1 to 2 MHz for the VCO in low-range mode
- Capability to work either in integer or fractional mode
- A 13-bit sigma-delta modulator, allowing to fine-tune the VCO frequency by steps of 11 to 0.3 ppm
- The sigma-delta modulator can be updated on-the-fly without generating frequency overshoots on PLLs outputs.
- Each PLL offers 3 outputs with post-dividers.

The PLLs are controlled via RCC_PLLxDIVR, RCC_PLLxFRACR, RCC_PLLCFGR and RCC_CR registers.

The frequency of the reference clock provided to the PLLs (refx_ck) must range from 1 to 16 MHz. The DIVMx dividers of the RCC PLLs clock source selection register (RCC_PLLCKSELR) must be properly programmed in order to match this condition. In addition, the PLLxRGE[1:0] field of the RCC PLLs configuration register (RCC_PLLCFGR) must be set according to the reference input frequency to guarantee an optimal performance of the PLL.

The user application can then configure the proper VCO. The smaller range (150-420 MHz) must be chosen when the reference clock frequency is lower to 2 MHz.

To reduce the power consumption, it is recommended to configure the VCO output to the smaller range.

DIVNx loop divider must be programmed to achieve the expected frequency at VCO output. In addition, the VCO output range must be respected.
The PLLs operate in integer mode when the value of SH_REG bit of the FRACNx shadow register is set to 0. The SH_REG bit is updated with the FRACNx value when PLLxFRACEN bit goes from 0 to 1. The sigma-delta modulator is designed in order to minimize the jitter impact while allowing very small frequency steps.

The PLLs can be enabled by setting PLLxON to 1. The PLLxRDY bits indicate that the PLL is ready (means locked).

**Note:** Before enabling the PLLs, make sure that the reference frequency (refx_ck) provided to the PLL is stable, so the hardware does not allow changing DIVMx when the PLLx is ON and it is also not possible to change PLLSRC when one of the PLL is ON.

The hardware prevents writing PLL1ON to 0 if the PLL1 is currently used to deliver the system clock. There are other hardware protections on the clock generators (refer to HSE oscillator on page 342, HSI oscillator on page 345 and CSI oscillator on page 346).

The following PLL parameters cannot be changed once the PLL is enabled: DIVNx, PLLxRGE, PLLxVCOSEL, DIVPx, DIVQx, DIVRx, DIVPxE, DIVQxEN and DIVRxEN.

To insure an optimal behavior of the PLL when one of the post-divider (DIVP, DIVQ or DIVR) is not used, the application must set the enable bit (DIVyEN) as well as the corresponding post-divider bits (DIVP, DIVQ or DIVR) to 0.

If the above rules are not respected, the PLL output frequency is not guaranteed.

**Output frequency computation**

When the PLL is configured in integer mode (SH_REG = 0), the VCO frequency \( F_{\text{VCO}} \) is given by the following expression:

\[
F_{\text{VCO}} = F_{\text{REF}_C} \times \text{DIVN}
\]

\[
F_{\text{PLL}_y\_CK} = \left( F_{\text{VCO}} \right) \div \left( \text{DIVy} + 1 \right) \quad \text{with } y = \text{P, Q or R}
\]

When the PLL is configured in fractional mode (SH_REG different from 0), the DIVN divider must be initialized before enabling the PLLs. However, it is possible to change the value of FRACNx on-the-fly without disturbing the PLL output.

This feature can be used either to generate a specific frequency from any crystal value with a good accuracy or to fine-tune the frequency on-the-fly.

For each PLL, the VCO frequency is given by the following formula:

\[
F_{\text{VCO}} = F_{\text{ref}_c} \times \left( \text{DIVN} + \frac{\text{FRACN}}{2^{13}} \right)
\]

**Note:** For PLL1, DIVP can only take odd values.
The PLLs are disabled by hardware when:
- the system enters Stop or Standby mode
- an HSE failure occurs when HSE or PLL (clocked by HSE) are used as system clock

PLL initialization phase

*Figure 51* shows the recommended PLL initialization sequence in integer and fractional mode. The PLLx are supposed to be disabled at the start of the initialization sequence:

1. Initialize the PLLs registers according to the required frequency.
   - Set PLLxFRACEN of *RCC PLLs configuration register (RCC_PLLCFGR)* to 0 for integer mode.
   - For fractional mode, set FRACN to the required initial value (FracInitValue) and then set PLLxFRACEN to 1.

2. Once the PLLxON bit is set to 1, the user application must wait until PLLxRDY bit is set to 1. If the PLLx is in fractional mode, the PLLxFRACEN bit must not be set back to 0 as long as PLLxRDY = 0.

3. Once the PLLxRDY bit is set to 1, the PLLx is ready to be used.

4. If the application intends to tune the PLLx frequency on-the-fly (possible only in fractional mode), then:
   a) PLLxFRACEN must be set to 0. When PLLxFRACEN = 0, the sigma-eelta modulator is still operating with the value latched into SH_REG.
   b) A new value must be uploaded into PLLxFRACR (FracValue(n)).
   c) PLLxFRACEN must be set to 1, in order to latch the content of PLLxFRACR into its shadow register.

*Note: When the PLLxRDY goes to 1, it means that the difference between the PLLx output frequency and the target value is lower than ±2%.*
Figure 51. PLLs Initialization flowchart

 PLL enable sequence
integer mode

 Select clock source (RCC_CKSELR)
- (PLLSRC)

 Init pre-divider (RCC_CKSELR)
- DIVMx

 PLLx config (RCC_PLLCFGR)
- PLLxVCSEL, PLLxRGE
- PLLxFRACEN = 0
- DIVPxEN, DIVQxEN, DIVRxEN
 Init PLLx dividers (RCC_PLLxDIVR)
- DIVNx, DIVPx, DIVQx, DIVRx

 Enable PLLx (RCC_CR)
- PLLxON = 1

 No
PLLxRDY = 1 ?

 Yes
Ready for use in integer mode

 PLL enable sequence
fractional mode

 Select clock source (RCC_CKSELR)
- (PLLSRC)

 Init pre-divider (RCC_CKSELR)
- DIVMx

 Init fractional value (RCC_PLLxFRACR)
- FRACN= FracInitValue
 PLLx config (RCC_PLLCFGR)
- PLLxVCSEL, PLLxRGE
- PLLxFRACEN = 1
- DIVPxEN, DIVQxEN, DIVRxEN
 Init PLLx dividers (RCC_PLLxDIVR)
- DIVNx, DIVPx, DIVQx, DIVRx

 Enable PLLx (RCC_CR)
- PLLxON = 1

 No
PLLxRDY = 1 ?

 Yes

 Ready for use in fractional mode

 Value update on-the-fly

 Disable fractional mode (RCC_PLLCFGR)
- PLLxFRACEN = 0
 Init fractional value (RCC_PLLxFRACR)
- FRACN= FracValue(n)
 Enable fractional mode (RCC_PLLCFGR)
- PLLxFRACEN = 1

 Ready for use in fractional mode
8.5.6 System clock (sys_ck)

System clock selection

After a system reset, the HSI is selected as system clock and all PLLs are switched OFF. When a clock source is used for the system clock, it is not possible for the software to disable the selected source via the xxxON bits.

Of course, the system clock can be stopped by the hardware when the system enters Stop or Standby mode.

When the system is running, the user application can select the system clock (sys_ck) among the 4 following sources:

- HSE
- HSI
- CSI
- pll1_p_ck

This function is controlled by programming the RCC clock configuration register (RCC_CFGR). A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source that is not yet ready is selected, the switch occurs when the clock source is ready.

The SWS status bits in the RCC clock configuration register (RCC_CFGR) indicate which clock is currently used as system clock. The other status bits in the RCC_CR register indicate which clock(s) is (are) ready.

System clock generation

*Figure 52* shows a simplified view of the clock distribution for the CPU and busses. All the dividers shown in the block diagram can be changed on-the-fly without generating timing violations. This feature is a very simple solution to adapt the busses frequencies to the application needs, thus optimizing the power consumption.

The CDCPRE divider can be used to adjust the CPU clock. However this also impacts the clock frequency of all bus matrix.

In the same way, HPRE divider can be used to adjust the clock for the CPU domain bus matrix, but this also impacts the clock frequency of bus matrix of SmartRun domain.

Most of the prescalers are controlled via RCC_CDCFGR and RCC_SRDCFGFR registers.
This block also provides the clock for the timers (\texttt{rcc\_timx\_ker\_ck} and \texttt{rcc\_timy\_ker\_ck}). The frequency of the timers clock depends on the APB prescaler corresponding to the bus to which the timer is connected, and on TIMPRE bit.

\textbf{Table 55} shows how to select the timer clock frequency.

\textbf{Table 55. Ratio between clock timer and pclk}

<table>
<thead>
<tr>
<th>CDPPRE1(1)</th>
<th>CDPPRE2</th>
<th>TIMPRE (2)</th>
<th>( F_{rcc_timx_ker_ck} )</th>
<th>( F_{rcc_timy_ker_ck} )</th>
<th>( F_{rcc_pclk1} )</th>
<th>( F_{rcc_pclk2} )</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0xx</td>
<td>( F_{rcc_hclk1} )</td>
<td>( F_{rcc_hclk1} )</td>
<td>( F_{rcc_hclk1} )</td>
<td>( F_{rcc_hclk1} )</td>
<td>The timer clock is equal to the bus clock.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>( F_{rcc_hclk1} )</td>
<td>( F_{rcc_hclk1} / 2 )</td>
<td>( F_{rcc_hclk1} / 2 )</td>
<td>( F_{rcc_hclk1} / 4 )</td>
<td>The timer clock is twice as fast as the bus clock.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101</td>
<td>( F_{rcc_hclk1} / 2 )</td>
<td>( F_{rcc_hclk1} / 4 )</td>
<td>( F_{rcc_hclk1} / 4 )</td>
<td>( F_{rcc_hclk1} / 8 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>110</td>
<td>( F_{rcc_hclk1} / 4 )</td>
<td>( F_{rcc_hclk1} / 8 )</td>
<td>( F_{rcc_hclk1} / 8 )</td>
<td>( F_{rcc_hclk1} / 16 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>111</td>
<td>( F_{rcc_hclk1} / 8 )</td>
<td>( F_{rcc_hclk1} / 16 )</td>
<td>( F_{rcc_hclk1} / 16 )</td>
<td>( F_{rcc_hclk1} / 32 )</td>
<td></td>
</tr>
</tbody>
</table>

(1) Can be changed on-the-fly. * Represents the selected mux input after a system reset.
8.5.7 Handling clock generators in Stop and Standby modes

When the whole system enters Stop mode, all the clocks (system and kernel clocks) are stopped as well as the following clock sources:

- CSI, HSI (depending on HSIKERON, and CSIKERON bits)
- HSE
- PLL1, PLL2 and PLL3
- HSI48

The content of the RCC registers is not altered except for PLL1ON, PLL2ON, PLL3ON HSEON and HSI48ON that are set to 0.

Exiting Stop mode

When the microcontroller exits system Stop mode via a wakeup event, the application can select which oscillator (HSI and/or CSI) is used to restart. The STOPWUCK bit selects the oscillator used as system clock. The STOPKERWUCK bit selects the oscillator used as kernel clock for peripherals. The STOPKERWUCK bit is useful if after a system Stop, a peripheral needs a kernel clock generated by an oscillator different from the one used for the system clock.

All these bits belong to the **RCC clock configuration register (RCC_CFGR)**.

**Table 56. STOPWUCK and STOPKERWUCK description**

<table>
<thead>
<tr>
<th>STOPWUCK</th>
<th>STOPKERWUCK</th>
<th>Activated oscillator when system exits Stop mode</th>
<th>Distributed clocks when system exits Stop mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>System clock</td>
<td>Kernel clock</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>HSI</td>
<td>HSI</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>HSI</td>
<td>HSI and/or CSI</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>CSI</td>
<td>CSI</td>
</tr>
</tbody>
</table>

1. CDPPRE1 and CDPPRE2 belong to **RCC CPU domain clock configuration register 2 (RCC_CDCFG1)**.
2. TIMPRE belongs to **RCC clock configuration register (RCC_CFGR)**.
During Stop mode

There are two specific cases where the HSI or CSI can be enabled during system Stop mode:

- when a dedicated peripheral requests the kernel clock
  The peripheral receives the HSI or CSI according to the kernel clock source selected for this peripheral (via PERxSRC).
- when the HSIKERON or CSIKERON bits are set
  The HSI and CSI are kept running during Stop mode but the outputs are gated. The clock is then available immediately when the system exits Stop mode or when a peripheral requests the kernel clock (see Table 57 for details).

HSIKERON and CSIKERON bits belong to RCC source control register (RCC_CR). Table 57 gives a detailed description of their behavior.

<table>
<thead>
<tr>
<th>HSIKERON (CSIKERON)</th>
<th>HSI (CSI) state during Stop mode</th>
<th>HSI (CSI) setting time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>→ OFF</td>
<td>( t_{su}^{(HSI)} ) ( t_{su}^{(CSI)} ) ( ^{(1)} )</td>
</tr>
<tr>
<td>1</td>
<td>→ Running and gated</td>
<td>Immediate</td>
</tr>
</tbody>
</table>

1. \( t_{su}^{(HSI)} \) and \( t_{su}^{(CSI)} \) are the startup times of the HSI and CSI oscillators (refer to the product datasheet for values of these parameters).

When the microcontroller exists system Standby mode, the HSI is selected as system and kernel clock. The RCC registers are reset to their initial values except for the RCC_RSR and RCC_BDCR registers.

Note that the HSI and CSI outputs provide two clock paths (see Figure 45):

- one path for the system clock (hsi_ck or csi_ck)
- one path for the peripheral kernel clock (hsi_ker_ck or csi_ker_ck).

When a peripheral requests the kernel clock in system Stop mode, only the path providing the hsi_ker_ck or csi_ker_ck is activated.

Caution: It is not guaranteed that the CPU gets automatically the same clock frequencies when leaving CStop mode: this mainly depends on the system state. For example if the CPU goes to CStop, while the SmartRun domain is kept in CRun, the clock settings remain unchanged when the CPU exits from CStop. If the SmartRun domain goes to CStop while the CPU is also in CStop, then the CPU operates with HSI or CSI when exiting from CStop mode.

8.5.8 Kernel clock selection

Some peripherals are designed to work with two different clock domains that operate asynchronously:

- a clock domain synchronous with the register and bus interface (ckg_bus_perx clock)
- a clock domain generally synchronous with the peripheral (kernel clock)
The benefit of having peripherals supporting these two clock domains is that the user application has more freedom to choose optimized clock frequency for the CPU, bus matrix and for the kernel part of the peripheral.

As a consequence, the user application can change the bus frequency without reprogramming the peripherals. As an example an on-going transfer with UART is not disturbed if its APB clock is changed on-the-fly.

*Table 58* shows the kernel clock that the RCC can deliver to the peripherals. Each row of this table represents a mux and the peripherals connected to its output. The columns starting from number 4 represents the clock sources. Column 3 gives the maximum allowed frequency at each mux output. It is up to the user to respect these requirements.

**Table 58. Kernel clock distribution overview**

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Clock mux control bits</th>
<th>Domain</th>
<th>Clock sources (^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>pl1 q-ck</td>
<td>pl1 p-ck</td>
</tr>
<tr>
<td>LTDC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FMC</td>
<td>FMCSSEL</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>OCTOSPI</td>
<td>OCTOSPISEL</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>SDMMC1</td>
<td>SDMMCSEL</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>SDMMC2</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DFSM1 Ack</td>
<td>SAI1SEL</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DFSM1 clk</td>
<td>DFSM1SEL</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FDCAN</td>
<td>FDCANSEL</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>HDMI-CEC</td>
<td>CECSEL</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I2C1,2,3</td>
<td>I2C123SEL</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LPTIM1</td>
<td>LPTIM1SEL</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>TIM[8:1],</td>
<td>TIM[17:12]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RNG</td>
<td>RNGSEL</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>SAI1(5)</td>
<td>SAI1SEL</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SAI2</td>
<td>SAI2ASEL</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
<td>2</td>
</tr>
</tbody>
</table>

\(^{(1)}\) Columns \(2, 3, 4\) and \(5\) correspond to the clock sources and the maximum allowed frequency at each mux output. It is up to the user to respect these requirements.
The table shows the kernel clock distribution overview for different peripherals and control bits, including:

- **Peripherals:** Various components like SPI, USART, USB, ADC, DAC, etc.
- **Clock mux control bits:** Initial selection after a system reset.
- **Domain:** CPU or SmartRun domain.
- **Clock sources:** Various options like pll1_q_ck, pll2_p_ck, etc.
- **Controlled by:** HSE, HSI, CSI, etc.

1. **X** represents the selected mux input after a system reset.
2. The bus clocks are the bus interface clocks to which the peripherals are connected. It can be APB, AHB or AXI clocks.
3. The per_ck clock can be hse_ck, hsi_ker_ck or csi_ker_ck according to CKPERSEL selection.
4. Clock CSI divided by 122.
5. The SAI1 max clock frequency can, in some cases, be limited by the constraints on DFSDM1 (see Figure 53).
6. With a duty cycle close to 50 %, meaning that DIV[P/Q/R]x values must be even. For SDMMCx, the duty cycle must be 50 % when supporting DDR.
7. The SPI6 maximum clock frequency can, in some cases, be limited by the constraints on DFSDM2 (see Figure 53 and Figure 54).
8. Clock HSE divided by RTCPRE.
Figure 53 to Figure 61 provide a more detailed description of kernel clock distribution. To simplify the drawings, the bus interface clocks (pclk, hclk) are not represented, even if they are gated with enable signals. Refer to Section 8.5.11: Peripheral clock gating control for more details.

To reduce the amount of switches, some peripherals share the same kernel clock source. Nevertheless, all peripherals have their dedicated enable signal.

Peripherals dedicated to audio applications

The audio peripherals generally need specific accurate frequencies, except for SPDIFRX. As shown in Figure 53 and Figure 56, the kernel clock of the SAI1 or SPI/I2S(s) can be generated by:

- PLL1 when the amount of active PLLs must be reduced (for SAI1 and SPI/I2S1 to 3)
- APB2 peripheral clock (for SPI/I2S4 and 5)
- APB4 peripheral clock (for SPI/I2S6)
- PLL2 or 3 for optimal flexibility in frequency generation
- HSE, HSI or CSI for use-cases where the current consumption is critical
- I2S_CKIN when an external clock reference needs to be used

Note: The SPDIFRX does not require a specific frequency but only a kernel clock frequency high enough to make the peripheral work properly. Refer to the SPDIFRX description for more details.

DFSDM1 can use the same clock as SAI1 and DFSDM2 can use the same clock as I2S6. This is useful when DFSDM1 and DFSDM2 are used for audio applications.

The SAI2 can use the symbol clock provided by the SPDIFRX. SPI/I2S1, 2, and 3 share the same kernel clock source. The SPI/I2S4 and 5 share the same kernel clock source (see Figure 54).
Figure 53. Kernel clock distribution for SAI1s, DFSDMs and SPDIFRX

1. X represents the selected mux input after a system reset.
2. This figure does not show the connection of the bus interface clock to the peripherals. For details on each enable cell, refer to Section 8.5.11: Peripheral clock gating control.
Peripherals dedicated to control and data transfer

Peripherals such as SPIs, I2Cs, UARTs do not need a specific kernel clock frequency but a clock fast enough to generate the correct baud rate, or the required bit clock on the serial interface. For that purpose the source can be selected among the following ones:

- PLL1 when the amount of active PLLs must be reduced
- PLL2 or PLL3 if better flexibility is required. As an example, this solution allows changing the frequency bus via PLL1 without affecting the speed of some serial interfaces.
- HSI or CSI for low-power use-cases or when the peripheral must quickly wake up from Stop mode (such as UART or I2C)

**Note:** UARTs also need the LSE clock when high baud rates are not required.

Figure 54. Kernel clock distribution for SPIs and SPI/I2S

1. X represents the selected mux input after a system reset.
2. This figure does not show the connection of the bus interface clock to the peripheral. For details on each enable cell, refer to Section 8.5.11: Peripheral clock gating control.
1. \( X \) represents the selected mux input after a system reset

2. This figure does not show the connection of the bus interface clock to the peripheral, for details on each enable cell, please refer to Section 8.5.11: Peripheral clock gating control.
1. X represents the selected mux input after a system reset.

2. This figure does not show the connection of the bus interface clock to the peripheral, for details on each enable cell, refer to Section 8.5.11: Peripheral clock gating control.

---

**Figure 56. Kernel clock distribution for UARTs, USARTs and LPUART1**

![Kernel clock distribution for UARTs, USARTs and LPUART1](image)

The switch is dynamic: the transition between two inputs is glitch-free.

---

**Figure 57. Kernel clock distribution for LTDC**

![Kernel clock distribution for LTDC](image)

The switch is dynamic: the transition between two inputs is glitch-free.

---

1. X represents the selected mux input after a system reset.

2. This figure does not show the connection of the bus interface clock to the peripheral. For details on each enable cell, refer to Section 8.5.11: Peripheral clock gating control.
The FMC, OCTOSPI1/2 and SDMMC1/2 can also use a clock different from the bus interface one for more flexibility.

**Figure 58. Kernel clock distribution for SDMMC, OCTOSPI and FMC**

The switch is dynamic: the transition between two inputs is glitch-free.

1. X represents the selected mux input after a system reset.
2. This figure does not show the connection of the bus interface clock to the peripheral. For details on each enable cell, refer to Section 8.5.11: Peripheral clock gating control.

**Figure 59** shows the clock distribution for the USB blocks. The USB1ULPI block receives its clock from the external PHY.

The USB1OTG block receives two kind of clocks:
- the clock for USB communications that can be selected among different sources thanks to the MUX controlled by USBSEL
- the isi_ck clock provided to the USB1OTG block in ADP mode on request.

**Figure 59. Kernel clock distribution for USB (2)**

The switch is dynamic: the transition between two inputs is glitch-free.

1. X represents the selected mux input after a system reset.
2. This figure does not show the connection of the bus interface clock to the peripheral. For details on each enable cell, refer to Section 8.5.11: Peripheral clock gating control.
1. X represents the selected mux input after a system reset.

2. This figure does not show the connection of the bus interface clock to the peripheral. For details on each enable cell, refer to Section 8.5.11: Peripheral clock gating control.

3. The ADC clock divider for the sys_ck_cpu do not need to be programmed but is set automatically depending on the HPRE (see also Figure 52: Core and bus clock generation).
**Figure 61. Kernel clock distribution for LPTIMs and HDMI-CEC (2)**

The switch is dynamic: the transition between two inputs is glitch-free.

1. X represents the selected mux input after a system reset
2. This figure does not show the connection of the bus interface clock to the peripheral. For details on each enable cell, refer to Section 8.5.11: Peripheral clock gating control.

**RTC/AWU clock**

The rtc_ck clock source can be one of the following:
- the hse_1M_ck (hse_ck divided by a programmable prescaler)
- the lse_ck
- the lsi_ck clock

The source clock is selected by programming the RTCSEL[1:0] bits in the RCC Backup domain control register (RCC_BDCR) and the RTCPRE[5:0] bits in the RCC clock configuration register (RCC_CFGR).

This selection cannot be modified without resetting the Backup domain.

If the LSE is selected as RTC clock, the RTC works normally even if the backup or the V_DD supply disappears.
The LSE clock is in the Backup domain, whereas the other oscillators are not. As a consequence:

- If LSE is selected as RTC clock, the RTC continues working even if the V_{DD} supply is switched OFF, provided the V_{BAT} supply is maintained.
- If LSI is selected as the RTC clock, the AWU state is not guaranteed if the V_{DD} supply is powered off.
- If the HSE clock is used as RTC clock, the RTC state is not guaranteed if the V_{DD} supply is powered off or if the V_{CORE} supply is powered off.

The \texttt{rtc\_ck} clock is enabled through RTCEN bit located in the \textit{RCC Backup domain control register (RCC\_BDCR)}.

The RTC bus interface clock (APB clock) is enabled through RTCAPBEN and RTCAPBLPEN bits located in RCC\_APB4ENR/LPENR registers.

\textbf{Note:}  To read the RTC calendar register when the APB clock frequency is less than seven times the RTC clock frequency (F_{APB} < 7 \times F_{RTCLK}), the software must read the calendar time and date registers twice. The data are correct if the second read access to RTC\_TR gives the same result than the first one. Otherwise a third read access must be performed.

\textbf{Watchdog clocks}

The RCC provides the clock for the two watchdog blocks available on the circuit. The independent watchdog (IWDG) is connected to the LSI. The window watchdog (WWDG) is connected to the APB clock.

If an independent watchdog is started by either hardware option or software access, the LSI is forced ON and cannot be disabled. After the LSI oscillator setup delay, the clock is provided to the IWDG.

\textbf{Clock frequency measurement using TIMx}

Most of the clock source generator frequencies can be measured by means of the input capture of TIMx.

- Calibrating the HSI or CSI with the LSE:
  The primary purpose of having the LSE connected to a TIMx input capture is to be able to accurately measure the HSI or CSI. This requires to use the HSI or CSI as system clock source either directly or via PLL1. The number of system clock counts between consecutive edges of the LSE signal gives a measurement of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm) we can determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process and/or temperature- and voltage-related frequency deviations.

  The basic concept consists in providing a relative measurement (e.g. HSI/LSE ratio). The precision is therefore tightly linked to the ratio between the two clock sources. The greater the ratio is, the more accurate the measurement is.

  The HSI and CSI oscillators have dedicated user-accessible calibration bits for this purpose (see \textit{RCC CSI calibration register (RCC\_CSICFGR)}). When HSI or CSI is used via the PLLx, the system clock can also be fine-tuned by using the fractional divider of the PLLs.

- Calibrating the LSI with the HSI:
  The LSI frequency can also be measured: this is useful for applications that do not have a crystal. The ultra-low-power LSI oscillator has a large manufacturing process
deviation. The LSI clock frequency can be measured using the more precise HSI clock source. Using this measurement, a more accurate RTC time base timeouts (when LSI is used as the RTC clock source) and/or an IWDG timeout with an acceptable accuracy can be obtained.

8.5.9 General clock concept overview

The RCC handles the distribution of the CPU, bus interface and peripheral clocks for the system (CPU and SmartRun domains), according to the operating mode of each function (refer to Section 8.5.1: Clock naming convention for details on clock definitions).

For each peripheral, the application can control the activation/deactivation of its kernel and bus interface clock. Prior to use a peripheral, the CPU must enable it (by setting PERxEN to 1), and define if this peripheral remains active in CSleep mode (by setting PERxLPEN to 1). This is called ‘allocation’ of a peripheral to the CPU (refer to Section 8.5.10: Peripheral allocation for more details).

The peripheral allocation is used by the RCC to automatically control the clock gating according to the CPU and domain modes, and by the PWR to control the supply voltages of CPU and SmartRun domains.

Figure 62 gives an example of peripheral allocation: the CPU enables SDMMC1, SPI5 and AHBSRAM1. The AXISRAMs, ITCM, DTCM1, DTCM2 and SRDSRAM are implicitly allocated to the CPU. The group composed of the CPU, bus matrix 1/2/3 and allocated peripherals makes up a sub-system (CPU_SS).

Note: The FLASH, AXISRAMs, ITCM, DTCM1, DTCM2, SRDSRAM, IWGD, IWGD, PWR, EXTI and RCC are common resources and are implicitly allocated to the CPU.
When the CPU domain enters CStop mode, the RCC automatically disables the bus interface and kernel clocks of all the peripherals of the CPU_SS, as well as the CPU clock. The PLLs, if enabled, are not disabled by the RCC since the SmartRun domain is still running.

The SmartRun domain can be kept in DRun mode while the CPU is in CStop mode and the CPU domain is in DStop or DStop2 mode. This is done by setting RUN_SRD bit in PWR_CPUCR register.

- If RUN_SRD is set to 1, then SmartRun domain is maintained in DRun mode, independently from the CPU modes (see power control, PWR_CPUCR register)
- If RUN_SRD is set to 0, then the SmartRun domain enters DStop mode when the CPU enters CStop mode (see Table 59).

Note that the CPU can control if the CPU domain is allowed to enter DStop or DStop2 modes or the SmartRun domain is allowed to enter in DStop when conditions are met, via bits RETDS_CD and PDDS_SRD of PWR control register (PWR_CPUCR).

A wakeup event is able to exit the CPU domain from DStop or DStop2 and the SmartRun domain from DStop mode.
In addition, more autonomy can be given to some peripherals located into the SmartRun domain (refer SmartRun domain autonomous mode).

**SmartRun domain autonomous mode**

The autonomous mode allows the delivery of the clocks to peripherals located in the SmartRun domain, even if the CPU is in CStop mode. When a peripheral has its autonomous bit activated, it receives its peripheral clocks according to the SmartRun domain state, if the CPU is in CStop mode:

- If the SmartRun domain is in DRun mode, peripherals with autonomous mode activated receive their peripheral clocks,
- If the SmartRun domain is in DStop mode, no peripheral clock is provided.

The autonomous mode does not prevent the SmartRun domain to enter DStop or DStop2 mode.

The autonomous bits are located in RCC SmartRun domain autonomous mode register (RCC_SRDAMR).

For example, the CPU can enter CStop mode while the I2S6 is filling the SRDSRAM with data received from an external device via BDMA2. When the amount of received data is reached, the CPU can be activated by a wakeup event. This can be done by setting the I2S6, the BDMA2, and SRDSRAM in autonomous mode, while keeping the SmartRun domain in DRun mode (RUN_SRD set to 1). In this example, the RCC does not switch off the PLLs as the SmartRun domain is always in DRun mode.

It is possible to go a step further with power-consumption reduction by combining the autonomous mode with the capability of some peripherals (UARTs, I2Cs) to request the kernel clock on their own, without waking-up the CPU. For example, if the system is expecting messages via I2C4, the whole system can be put in Stop mode. When the I2C4 peripheral detects a START bit, it generates a “kernel clock request”. This request enables the HSI or CSI, and a kernel clock is provided only to the requester (in our example the I2C4). The I2C4 then decodes the incoming message.

Several cases are then possible:

- If the device address of the message does not match, then I2C4 releases its “kernel clock request” until a new START condition is detected.
- If the device address of the incoming message matches, it must be stored into the SmartRun local memory. I2C4 is able to generate a wakeup event on address match to switch the SmartRun domain to DRun mode. The message is then transferred into memory via BDMA2, and the SmartRun domain go back to DStop mode without any CPU activation. Note that if the amount of data transferred into memory reached the transfer count, the BDMA2 can also generate an interrupt to wake-up the CPU.
- If the device address of the incoming message matches and the peripheral is setup to wake up the CPU, then I2C4 generates a wakeup event to activate the CPU.

Refer to the description of EXTI block to see which peripheral is able to perform a wakeup event to which domain.
Memory handling

The CPU can access all the memory areas available in the product:
- AXISRAM1, AXISRAM2, AXISRAM3, ITCM, DTCM1, DTCM2 and FLASH
- AHBSRAM1 and AHBSRAM2
- SRDSRAM and BKPRAM

As shown in Figure 62, FLASH, AXISRAMs, SRDSRAM, ITCM, DTCM1 and DTCM2 are implicitly allocated to the CPU. As a result, there is no enable bit allowing the CPU to allocate these memories.

The BKPRAM has a dedicated enable in order to gate the bus interface clock. The CPU needs to enable the BKPRAM prior to use it.

Note: The memory interface clocks (Flash memory and RAM interfaces) can be stopped by software during CSleep mode (via SRAMyLPEN bits).

Refer to Section 8.5.11: Peripheral clock gating control and Section 8.5.12: CPU and bus matrix clock gating control sections for details on clock enabling.

System states overview

Table 59 gives an overview of the system states with respect to the CPU and SmartRun domain modes.

- The system remains in Run mode as long as the SmartRun domain is in DRun mode. Several sub-states of system Run exist that are not detailed here (refer Section 6: Power control (PWR) for more information).
- The SmartRun domain can run while the CPU domain is in DStop or DStop2 mode thanks to RUN_SRD bits of PWR_CPUCR registers or when the SmartRun domain is in autonomous mode.
- The system remains in Stop mode as long as the SmartRun domain is in DStop mode. This means implicitly that the CPU domain is in DStop or DStop2 mode. As soon as the CPU domain exits DStop or DStop2, the SmartRun domain switches to DRun mode.
- The system remains in Standby mode as long as CPU and SmartRun domains are in DStandby.
- Domain states versus CPU states:
  - When the CPU domain is in DRun mode, it means that its bus matrix is clocked and the CPU is in CRun mode.
  - When the CPU domain is in DStop mode, it means that its bus matrix is no longer clocked and the CPU is in CStop mode.
  - When a domain is in DStandby mode, it means that the domain including its CPU are powered down.

<table>
<thead>
<tr>
<th>System state</th>
<th>CPU domain state</th>
<th>SmartRun domain state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>DRun</td>
<td>DRun</td>
</tr>
<tr>
<td></td>
<td>DStop/DStop2</td>
<td></td>
</tr>
<tr>
<td>Stop</td>
<td>DStop/DStop2</td>
<td>DStop</td>
</tr>
<tr>
<td>Standby</td>
<td>DStandby</td>
<td>DStandby</td>
</tr>
</tbody>
</table>
8.5.10 Peripheral allocation

The CPU can allocate a peripheral and hence control its kernel and bus interface clock. The CPU can allocate a peripheral by setting the dedicated PERxEN bit located into the RCC_xxxxENR registers.

The CPU can control the peripheral clocks gating when it is in CSleep mode via the PERxLPEN bits located into the RCC_xxxxLPENR registers.

Refer to RCC source control register (RCC_CR) for additional information.

The peripheral allocation bits (PERxEN bits) are used by the hardware to provide the kernel and bus interface clocks to the peripherals. However they are also used to link peripherals to the CPU (CPU sub-system). In this way, the hardware is able to safely gate the peripheral clocks and bus matrix clocks according to CPU states. The PWR block also uses this information to control properly the domain states.

Clock switches and gating

- Clock switching delays
  The input selected by the kernel clock switches can be changed dynamically without generating spurs or timing violation. As a consequence, switching from the original to the new input can only be performed if a clock is present on both inputs. If it is not the case, no clock is provided to the peripheral. To recover from this situation, the user must provide a valid clock to both inputs.
  
  During the transition from one input to another, the kernel clock provided to the peripheral is gated, in the worst case, during 2 clock cycles of the previously selected clock, and 2 clock cycles of the new selected clock. As shown in Figure 63, both input clocks must be present during transition time.

  \[\text{Figure 63. Kernel clock switching}\]

  \[
  \begin{array}{c}
  \text{PERxSEL} \\
  \text{in0}_{\text{ck}} \\
  \text{in1}_{\text{ck}} \\
  \text{rcc}_{\text{perx}}_{\text{ker}}_{\text{ck}} \\
  \end{array}
  \]

  \[\text{(Kernel clock provided to PERx)}\]

  \[
  \begin{array}{c}
  \text{Transition time} \\
  1 \\
  2 \\
  3 \\
  4 \\
  \end{array}
  \]

- Clock enabling delays
  In the same way, the clock gating logic synchronizes the enable command (coming generally from a kernel clock request or PERxEN bits) with the selected clock, in order to avoid generation of spurs.
    - A maximum delay of two periods of the enabled clock may occur between the enable command and the first rising edge of the clock. The enable command can
be the rising edge of the PERxEN bits of RCC_xxxxENR registers, or a kernel clock request asserted by a peripheral.

– A maximum delay of 1.5 periods of the disabled clock may occur between the disable command and the last falling edge of the clock. The disable command can be the falling edge of the PERxEN bits of RCC_xxxxENR registers, or a kernel clock request released by a peripheral.

**Note:** *Both the kernel and bus interface clocks are affected by this re-synchronization delay.*

### 8.5.11 Peripheral clock gating control

As mentioned previously, each peripheral requires a bus interface clock, named `rcc_perx_bus_ck` (for peripheral ‘x’). This clock can be an APB, AHB or AXI clock, according to which bus the peripheral is connected.

The clocks used as bus interface for peripherals located in the CPU domain, can be `rcc_aclk`, `rcc_hclk3` or `rcc_pclk3`, depending on the bus connected to each peripheral. For simplicity sake, these clocks are named `rcc_bus_cd_ck`.

Similarly, the signal `rcc_bus_srd_ck` represents `rcc_hclk4` or `rcc_pclk4` for peripherals located in the SmartRun domain.

Some peripherals (such as SAI or UART) also require a dedicated clock for their communication interface. This clock is generally asynchronous with respect to the bus interface clock. It is named kernel clock (`perx_ker_ckreq`). Both clocks can be gated according to several conditions detailed hereafter.

As shown in Figure 64, enabling the kernel and bus interface clocks of each peripheral depends on several input signals:

- PERxEN and PERxLPEN bits,
  PERxEN represents the peripheral enable (allocation) bit for the CPU. The CPU can write these bits to 1 via the RCC_xxxxENR registers.
- PERxAMEN bits belonging to *RCC SmartRun domain autonomous mode register (RCC_SRDAMR).*
- CPU state (`cpu_sleep` and `cpu_deepsleep` signals)
- SmartRun domain state (`srd_deepsleep` signal)
- Kernel clock request (`perx_ker_ckreq`) of the peripheral itself, when the feature is available

Refer to *Section 8.5.10: Peripheral allocation* for more details.
**Table 60** gives a detailed description of the enabling logic of the peripheral clocks for peripherals located in the CPU domain and allocated by the CPU.

### Table 60. Peripheral clock enabling for the CPU domain peripherals

<table>
<thead>
<tr>
<th>PERxEN</th>
<th>PERxLPEN</th>
<th>PERxSEL</th>
<th>perx_ker_ckreq</th>
<th>CPU state</th>
<th>rcc_perx_ker_cpu_en</th>
<th>rcc_perx_bus_cd_en</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>No clock provided to the peripheral, because PERxEN=0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>CRun</td>
<td>1</td>
<td>1</td>
<td>Kernel and bus interface clocks are provided to the peripheral, because the CPU is in CRun, and PERxEN = 1.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>CSleep</td>
<td>0</td>
<td>0</td>
<td>No clock provided to the peripheral, because the CPU is in CSleep and PERxLPEN = 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Kernel and bus interface clocks are provided to the peripheral, because CPU is in CSleep, and PERxLPEN = 1</td>
</tr>
</tbody>
</table>
As a summary, we can state that the kernel clock is provided to the peripherals located on the CPU domain when the following conditions are met:

1. The CPU is in CRun mode, and the peripheral is allocated.
2. The CPU is in CSleep mode and the peripheral is allocated with PERxLPEN = 1.
3. The CPU is in CStop mode, the peripheral is allocated with PERxLPEN = 1, the peripheral generates a kernel clock request and the selected clock is hsi_ker_ck or csi_ker_ck.
4. The CPU is in CStop mode, the peripheral is allocated with PERxLPEN = 1 and the kernel source clock of the peripheral is lse_ck or lsi_ck.

The bus interface clock is provided to the peripherals only when conditions 1 or 2 are met.
Table 61 gives a detailed description of the enabling logic of the kernel clock for all peripherals located in the SmartRun domain.

### Table 61. Peripheral clock enabling for SmartRun domain peripherals

<table>
<thead>
<tr>
<th>( \text{PERxEN} )</th>
<th>( \text{PERxLPEN} )</th>
<th>( \text{PERxAMEN} )</th>
<th>( \text{PERxSEL} )</th>
<th>( \text{CPU state} )</th>
<th>( \text{SmartRun domain state} )</th>
<th>( \text{rcc_perx_ker_srd_en} )</th>
<th>( \text{rcc_perx_bus_srd_en} )</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 X X X X X X</td>
<td>Any</td>
<td>Any</td>
<td>0 0</td>
<td>No clock provided to the peripheral, as ( \text{PERxEN} = 0 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 X X X X X X</td>
<td>CRun</td>
<td>1 1</td>
<td>Kernel and bus interface clocks are provided to the peripheral, because the CPU is in CRun, and ( \text{PERxEN} = 1 ).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 X X X X</td>
<td>CSleep</td>
<td>0 0</td>
<td>No clock provided to the peripheral, because the CPU is in CSleep, and ( \text{PERxLPEN} = 0 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 X X X X</td>
<td>Run</td>
<td>1 1</td>
<td>Kernel and bus interface clocks are provided to the peripheral, because the CPU is in CSleep, and ( \text{PERxLPEN} = 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 X 0 X X X</td>
<td>CStop</td>
<td>0 0</td>
<td>As the CPU is in CStop, and ( \text{PERxEN}=1 ), then the kernel clock gating depends on the SmartRun domain state and ( \text{PERxAMEN} ) bits. No clock provided to the peripheral because ( \text{PERxAMEN} = 0 ).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 X 1 X X X</td>
<td>not lse_ck and not lsi_ck 0</td>
<td>1 1</td>
<td>The kernel and bus interface clocks are provided because even if the CPU is in CStop mode, the SmartRun domain is in Run mode, with ( \text{PERxEN} ) and ( \text{PERxAMEN} ) bits set to 1.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 X 1 X X X</td>
<td>Stop</td>
<td>0 0</td>
<td>No clock provided to the peripheral, because the SmartRun domain is in Stop, ( \text{req_ker_perx} = 0 ), and ( \text{lse_ck} ) or ( \text{lsi_ck} ) is not selected</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
As a summary, we can state that the kernel clock is provided to the peripherals of the SmartRun domain if the following conditions are met:

1. The CPU is in CRun mode and the peripheral is allocated.
2. The CPU is in CSleep mode and the peripheral is allocated with PERxLPEN = 1.
3. The CPU is in CStop mode, the peripheral is allocated and the SmartRun domain is in Run mode with PERxAMEN = 1.
4. The CPU is in CStop mode, the peripheral is allocated, the SmartRun domain is in Stop mode with PERxAMEN = 1, the peripheral is generating a kernel clock request and the kernel clock source is \textit{hsi\_ker\_ck} or \textit{csi\_ker\_ck}.
5. The CPU is in CStop mode, the peripheral is allocated, the SmartRun domain is in Stop mode with PERxAMEN = 1 and the kernel clock source of the peripheral is \textit{lse\_ck} or \textit{lsi\_ck}.

The bus interface clock is provided to the peripherals only when condition 1, 2 or 3 is met.
**Note:** When they are set to 1, the autonomous bits indicate that the associated peripheral receives a kernel clock according to the SmartRun domain state and not according to the mode of the CPU.

Only I2C, U(S)ART and LPUART peripherals are able to request the kernel clock. This feature gives to the peripheral the capability to transfer data with an optimal power consumption.

The autonomous bits dedicated to some peripherals located in the SmartRun domain, allow the data transfer with external devices without activating the CPU.

In order for the LPTIMER to operate with `lse_ck` or `lsi_ck` when the circuit is in Stop mode, the user application must select the `lsi_ck` or `lse_ck` input via LPTIMxSEL fields, and set LPTIMxAMEN and LPTIMxLPEN bits to 1.

### 8.5.12 CPU and bus matrix clock gating control

For each domain it is possible to control the activation/deactivation of the CPU clock and bus matrix clock.

For information about convention naming, refer to [Section 8.5.11: Peripheral clock gating control](#).

The clocks of the CPU, AHB and AXI bridges and APB busses are enabled according to the rules hereafter:

- The CPU clock `rcc_cpu_ck` is enabled when the CPU is in CRun mode.
- The AXI bridge clock is enabled when the CPU domain is in DRun mode.
- Each AXI master and slave has an independent clock gating that can be activated to further reduce the power consumption (register RCC_CKGAENR). When activated the clock is automatically enabled on bus transaction request.
- The CPU domain AHB bridges clocks are enabled when the CPU domain is in DRun mode.
- The SmartRun domain AHB bridge clock is enabled when one of the following conditions is met:
  - The CPU is in CRun or CSleep mode.
  - When the RUN_SRD bit is set to 1, independently of CPU modes.
  - When the `SRD_deepsleep` signal is inactive (0), independently of CPU modes.
- The APB1,2,3 busses are enabled when one of the following conditions is met:
  - The CPU is in CRun mode.
  - When the CPU is in CSSleep mode with at least one peripheral connected to this bus having both its PERxEN and PERxLPEN set to 1.
- The APB4 bus is enabled when the SmartRun domain is in DRun mode.

As shown in the [Figure 65](#), the enabling of the core and bus clock of each domain depends on several input signals:

- `cpu_sleep` and `cpu_deepsleep` signals from the CPU
- `srdf_deepsleep` signal
- RCC_xxxxENR.PERxEN bits of peripherals located on the CPU domain
8.6 RCC interrupts

The RCC provides three interrupt lines:

- **rcc_it**: general interrupt line providing events when the PLLs are ready or when the oscillators are ready
- **rcc_hsecss_it**: interrupt line dedicated to the failure detection of the HSE CSS (clock security system)
- **rcc_lsecss_it**: interrupt line dedicated to the failure detection of the LSE CSS

The interrupt enable is controlled via **RCC clock source interrupt enable register (RCC_CIER)**, except for the HSE CSS failure. When the HSE CSS feature is enabled, it not possible to mask the interrupt generation.

The interrupt flags can be checked via **RCC clock source interrupt flag register (RCC_CIFR)**, and those flags can be cleared via **RCC clock source interrupt clear register (RCC_CICR)**.

**Note:** The interrupt flags are not relevant if the corresponding interrupt enable bit is not set.

**Table 62** gives a summary of the interrupt sources and the way to control them.
### 8.7 RCC registers

#### 8.7.1 RCC source control register (RCC_CR)

**Address offset:** 0x000  
**Reset value:** 0x0000 0025

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

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

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSIDIVF</td>
<td>CSIF</td>
<td>CSIRDY</td>
<td>CSION</td>
<td>CSIRDYF</td>
<td>CSIRDY</td>
<td>CSION</td>
<td>CSIF</td>
<td>HOV1(10)</td>
<td>HSI1</td>
<td>HSI0</td>
<td>HSI0</td>
<td>HSI0</td>
<td>HSI0</td>
<td>HSI0</td>
<td>HSI0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

1. The security system feature must also be enabled (LSECSSON = 1), in order to generate interrupts.  
2. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = 1).
Bits 31:30  Reserved, must be kept at reset value.

Bit 29  **PLL3RDY**: PLL3 clock ready flag
   Set by hardware to indicate that the PLL3 is locked.
   0: PLL3 unlocked (default after reset)
   1: PLL3 locked

Bit 28  **PLL3ON**: PLL3 enable
   Set and cleared by software to enable PLL3.
   Cleared by hardware when entering Stop or Standby mode.
   0: PLL3 OFF (default after reset)
   1: PLL3 ON

Bit 27  **PLL2RDY**: PLL2 clock ready flag
   Set by hardware to indicate that the PLL2 is locked.
   0: PLL2 unlocked (default after reset)
   1: PLL2 locked

Bit 26  **PLL2ON**: PLL2 enable
   Set and cleared by software to enable PLL2.
   Cleared by hardware when entering Stop or Standby mode.
   0: PLL2 OFF (default after reset)
   1: PLL2 ON

Bit 25  **PLL1RDY**: PLL1 clock ready flag
   Set by hardware to indicate that the PLL1 is locked.
   0: PLL1 unlocked (default after reset)
   1: PLL1 locked

Bit 24  **PLL1ON**: PLL1 enable
   Set and cleared by software to enable PLL1.
   Cleared by hardware when entering Stop or Standby mode. Note that the hardware prevents writing this bit to 0, if the PLL1 output is used as the system clock.
   0: PLL1 OFF (default after reset)
   1: PLL1 ON

Bits 23:21  Reserved, must be kept at reset value.

Bit 20  **HSEEXT**: external high speed clock type in Bypass mode
   Set and reset by software to select the external clock type (analog or digital).
   The external clock must be enabled with the HSEON bit to be used by the device.
   The HSEEXT bit can be written only if the HSE oscillator is disabled.
   0: HSE in analog mode (default after reset)
   1: HSE in digital mode

Bit 19  **HSECSSON**: HSE clock security system enable
   Set by software to enable clock security system on HSE.
   This bit is “set only” (disabled by a system reset or when the system enters in Standby mode). When HSECSSON is set, the clock detector is enabled by hardware when the HSE is ready and disabled by hardware if an oscillator failure is detected.
   0: CSS on HSE OFF (clock detector OFF) (default after reset)
   1: CSS on HSE ON (clock detector ON if the HSE oscillator is stable, OFF if not).
Bit 18  **HSEBYP**: HSE clock bypass
Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit to be used by the device.
The HSEBYP bit can be written only if the HSE oscillator is disabled.
0: HSE oscillator not bypassed (default after reset)
1: HSE oscillator bypassed with an external clock

Bit 17  **HSERDY**: HSE clock ready flag
Set by hardware to indicate that the HSE oscillator is stable.
0: HSE clock is not ready (default after reset)
1: HSE clock is ready

Bit 16  **HSEON**: HSE clock enable
Set and cleared by software.
Cleared by hardware to stop the HSE when entering Stop or Standby mode.
This bit cannot be cleared if the HSE is used directly (via SW mux) as system clock, or if the HSE is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1).
0: HSE is OFF (default after reset)
1: HSE is ON

Bit 15  **CDCKRDY**: CPU domain clocks ready flag
Set by hardware to indicate that the following CPU domain clocks are available: APB1, APB2, AHB bus matrix.
0: CPU domain clocks are not available (default after reset)
1: CPU domain clocks are available

Bit 14  **CPUCKRDY**: CPU related clocks ready flag
Set by hardware to indicate that the CPU related clocks (CPU, APB3, AXI bus matrix and related memories) are available.
0: CPU related clocks are not available (default after reset)
1: CPU related clocks are available

Bit 13  **HSI48RDY**: HSI48 clock ready flag
Set by hardware to indicate that the HSI48 oscillator is stable.
0: HSI48 clock is not ready (default after reset)
1: HSI48 clock is ready

Bit 12  **HSI48ON**: HSI48 clock enable
Set by software and cleared by software or by the hardware when the system enters to Stop or Standby mode.
0: HSI48 is OFF (default after reset)
1: HSI48 is ON

Bits 11:10  Reserved, must be kept at reset value.

Bit 9  **CSIKERON**: CSI clock enable in Stop mode
Set and reset by software to force the CSI to ON, even in Stop mode, in order to be quickly available as kernel clock for some peripherals. This bit has no effect on the value of CSION.
0: no effect on CSI (default after reset)
1: CSI is forced to ON even in Stop mode

Bit 8  **CSIRDY**: CSI clock ready flag
Set by hardware to indicate that the CSI oscillator is stable. This bit is activated only if the RC is enabled by CSION (it is not activated if the CSI is enabled by CSIKERON or by a peripheral request).
0: CSI clock is not ready (default after reset)
1: CSI clock is ready
Bit 7  **CSION**: CSI clock enable
Set and reset by software to enable/disable CSI clock for system and/or peripheral.
Set by hardware to force the CSI to ON when the system leaves Stop mode, if STOPWUCK = 1 or STOPKERWUCK = 1.
This bit cannot be cleared if the CSI is used directly (via SW mux) as system clock, or if the CSI is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1).
0: CSI is OFF (default after reset)
1: CSI is ON

Bit 6  Reserved, must be kept at reset value.

Bit 5  **HSIDIVF**: HSI divider flag
Set and reset by hardware.
As a write operation to HSIDIV has not an immediate effect on the frequency, this flag indicates the current status of the HSI divider. HSIDIVF goes immediately to 0 when HSIDIV value is changed, and is set back to 1 when the output frequency matches the value programmed into HSIDIV.
0: new division ratio not yet propagated to hsi(_ker)_ck (default after reset)
1: hsi(_ker)_ck clock frequency reflects the new HSIDIV value (default register value when the clock setting is completed)

Bits 4:3  **HSIDIV[1:0]**: HSI clock divider
Set and reset by software.
These bits allow selecting a division ratio in order to configure the wanted HSI clock frequency. The HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to 1). In that case, the new HSIDIV value is ignored.
00: division by 1, hsi(_ker)_ck = 64 MHz (default after reset)
01: division by 2, hsi(_ker)_ck = 32 MHz
10: division by 4, hsi(_ker)_ck = 16 MHz
11: division by 8, hsi(_ker)_ck = 8 MHz

Bit 2  **HSIRDY**: HSI clock ready flag
Set by hardware to indicate that the HSI oscillator is stable.
0: HSI clock is not ready (default after reset)
1: HSI clock is ready

Bit 1  **HSIKERON**: HSI clock enable in Stop mode
Set and reset by software to force the HSI to ON, even in Stop mode, in order to be quickly available as kernel clock for peripherals. This bit has no effect on the value of HSION.
0: no effect on HSI (default after reset)
1: HSI is forced to ON even in Stop mode

Bit 0  **HSION**: HSI clock enable
Set and cleared by software.
Set by hardware to force the HSI to ON when the product leaves Stop mode, if STOPWUCK = 0 or STOPKERWUCK = 0.
Set by hardware to force the HSI to ON when the product leaves Standby mode or in case of a failure of the HSE which is used as the system clock source.
This bit cannot be cleared if the HSI is used directly (via SW mux) as system clock, or if the HSI is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1).
0: HSI is OFF
1: HSI is ON (default after reset)
8.7.2 RCC HSI calibration register (RCC_HSICFGR)

Address offset: 0x004
Reset value: 0x4000 0XXX
Reset value depends on the flash option bytes setting.

<table>
<thead>
<tr>
<th>Bit</th>
<th>HSITRIM[6:0]</th>
<th>HSICAL[11:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rw</td>
<td>r</td>
</tr>
<tr>
<td>30</td>
<td>rw</td>
<td>r</td>
</tr>
<tr>
<td>29</td>
<td>rw</td>
<td>r</td>
</tr>
<tr>
<td>28</td>
<td>rw</td>
<td>r</td>
</tr>
<tr>
<td>27</td>
<td>rw</td>
<td>r</td>
</tr>
<tr>
<td>26</td>
<td>rw</td>
<td>r</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>r</td>
</tr>
</tbody>
</table>

Bit 31  Reserved, must be kept at reset value.

Bits 30:24 HSITRIM[6:0]: HSI clock trimming
Set by software to adjust calibration.
HSITRIM field is added to the engineering option bytes loaded during reset phase
(FLASH_HSI_opt) in order to form the calibration trimming value.
HSICAL = HSITRIM + FLASH_HSI_opt.

Note: The reset value of the field is 0x40.

Bits 23:12  Reserved, must be kept at reset value.

Bits 11:0 HSICAL[11:0]: HSI clock calibration
Set by hardware by option byte loading during system reset nreset.
Adjusted by software through trimming bits HSITRIM.
This field represents the sum of engineering option byte calibration value and HSITRIM bits value.

8.7.3 RCC clock recovery RC register (RCC_CRRCR)

Address offset: 0x008
Reset value: 0x0000 0XXX
Reset value depends on the flash option bytes setting.

<table>
<thead>
<tr>
<th>Bit</th>
<th>HS48CAL[9:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>r</td>
</tr>
<tr>
<td>30</td>
<td>r</td>
</tr>
<tr>
<td>29</td>
<td>r</td>
</tr>
<tr>
<td>28</td>
<td>r</td>
</tr>
<tr>
<td>27</td>
<td>r</td>
</tr>
<tr>
<td>26</td>
<td>r</td>
</tr>
<tr>
<td>25</td>
<td>r</td>
</tr>
<tr>
<td>24</td>
<td>r</td>
</tr>
<tr>
<td>23</td>
<td>r</td>
</tr>
<tr>
<td>22</td>
<td>r</td>
</tr>
<tr>
<td>21</td>
<td>r</td>
</tr>
<tr>
<td>20</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>r</td>
</tr>
<tr>
<td>18</td>
<td>r</td>
</tr>
<tr>
<td>17</td>
<td>r</td>
</tr>
<tr>
<td>16</td>
<td>r</td>
</tr>
</tbody>
</table>

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

Bits 9:0 HS48CAL[9:0]: Internal RC 48 MHz clock calibration
Set by hardware by option byte loading during system reset nreset.
Read-only.
8.7.4 **RCC CSI calibration register (RCC_CSICFGR)**

Address offset: 0x00C
Reset value: 0x2000 0XXX
Reset value depends on the flash option bytes setting.

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

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

Bits 29:24 **CSITRIM[5:0]**: CSI clock trimming
Set by software to adjust calibration.
CSITRIM field is added to the engineering option bytes loaded during reset phase (FLASH_CSI_opt) in order to form the calibration trimming value.
CSICAL = CSITRIM + FLASH_CSI_opt.

*Note: The reset value of the field is 0x20.*

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:0 **CSICAL[7:0]**: CSI clock calibration
Set by hardware by option byte loading during system reset nreset.
Adjusted by software through trimming bits CSITRIM.
This field represents the sum of engineering option byte calibration value and CSITRIM bits value.
8.7.5 RCC clock configuration register (RCC_CFRG)

Address offset: 0x010
Reset value: 0x0000 0000

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

Bits 31:29 **MCO2SEL[2:0]:** microcontroller clock output 2
Set and cleared by software. Clock source selection may generate glitches on MCO2.
It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.
000: system clock selected (sys_ck) (default after reset)
001: PLL2 oscillator clock selected (pll2_p_ck)
010: HSE clock selected (hse_ck)
011: PLL1 clock selected (pll1_q_ck)
100: CSI clock selected (csi_ck)
101: LSI clock selected (lsi_ck)
others: reserved

Bits 28:25 **MCO2PRE[3:0]:** MCO2 prescaler
Set and cleared by software to configure the prescaler of the MCO2. Modification of this prescaler may generate glitches on MCO2. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.
0000: prescaler disabled (default after reset)
0001: division by 1 (bypass)
0010: division by 2
0011: division by 3
0100: division by 4
...
1111: division by 15

Bits 24:22 **MCO1SEL[2:0]:** microcontroller clock output 1
Set and cleared by software. Clock source selection may generate glitches on MCO1.
It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.
000: HSI clock selected (hsi_ck) (default after reset)
001: LSE oscillator clock selected (lse_ck)
010: HSE clock selected (hse_ck)
011: PLL1 clock selected (pll1_q_ck)
100: HSI48 clock selected (hsi48_ck)
others: reserved
Bits 21:18 **MCO1PRE[3:0]**: MCO1 prescaler

Set and cleared by software to configure the prescaler of the MCO1. Modification of this prescaler may generate glitches on MCO1. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.

- 0000: prescaler disabled (default after reset)
- 0001: division by 1 (bypass)
- 0010: division by 2
- 0011: division by 3
- 0100: division by 4
  ...
- 1111: division by 15

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **TIMPRE**: timers clocks prescaler selection

This bit is set and reset by software to control the clock frequency of all the timers connected to APB1 and APB2 domains.

- 0: The timers kernel clock is equal to \( rcc\_hclk1 \) if CDPPRE1 or CDPPRE2 corresponds to a division by 1 or 2, else it is equal to \( 2 \times Frcc\_pclk1 \) or \( 2 \times Frcc\_pclk2 \) (default after reset)
- 1: The timers kernel clock is equal to \( 2 \times Frcc\_pclk1 \) or \( 2 \times Frcc\_pclk2 \) if CDPPRE1 or CDPPRE2 corresponds to a division by 1, 2 or 4, else it is equal to \( 4 \times Frcc\_pclk1 \) or \( 4 \times Frcc\_pclk2 \).

Refer to **Table 55: Ratio between clock timer and pclk** for more details.

Bits 14 Reserved, must be kept at reset value.

Bits 13:8 **RTCPRE[5:0]**: HSE division factor for RTC clock

Set and cleared by software to divide the HSE to generate a clock for RTC.

Caution: The software must set these bits correctly to ensure that the clock supplied to the RTC is lower than 1 MHz. These bits must be configured if needed before selecting the RTC clock source.

- 000000: no clock (default after reset)
- 000001: no clock
- 000010: HSE/2
- 000011: HSE/3
- 000100: HSE/4
  ...
- 111110: HSE/62
- 111111: HSE/63

Bit 7 **STOPKERWUCK**: kernel clock selection after a wake up from system Stop

Set and reset by software to select the kernel wakeup clock from system Stop.

- 0: HSI selected as wake up clock from system Stop (default after reset)
- 1: CSI selected as wake up clock from system Stop

See **Section 8.5.7: Handling clock generators in Stop and Standby modes** for details.
Bit 6 **STOPWUCK**: system clock selection after a wake up from system Stop
   Set and reset by software to select the system wakeup clock from system Stop.
   The selected clock is also used as emergency clock for the clock security system (CSS) on HSE.
   0: HSI selected as wake up clock from system Stop (default after reset)
   1: CSI selected as wake up clock from system Stop
   See Section 8.5.7: Handling clock generators in Stop and Standby modes for details.

Caution: STOPWUCK must not be modified when CSS is enabled (by HSECSSON bit) and the system clock is HSE (SWS = 10) or a switch on HSE is requested (SW =10).

Bits 5:3 **SWS[2:0]**: system clock switch status
   Set and reset by hardware to indicate which clock source is used as system clock.
   000: HSI used as system clock (hsi_ck) (default after reset)
   001: CSI used as system clock (csi_ck)
   010: HSE used as system clock (hse_ck)
   011: PLL1 used as system clock (pll1_p_ck)
   others: reserved

Bits 2:0 **SW[2:0]**: system clock and trace clock switch
   Set and reset by software to select system clock and trace clock sources (sys_ck and traceclk).
   Set by hardware in order to:
   – force the selection of the HSI or CSI (depending on STOPWUCK selection) when leaving a system Stop mode
   – force the selection of the HSI in case of failure of the HSE when used directly or indirectly as system clock
   000: HSI selected as system clock (hsi_ck) (default after reset)
   001: CSI selected as system clock (csi_ck)
   010: HSE selected as system clock (hse_ck)
   011: PLL1 selected as system clock (pll1_p_ck for sys_ck, pll1_r_ck for traceclk)
   others: reserved

### 8.7.6 RCC CPU domain clock configuration register 1 (RCC_CDCFGR1)

Address offset: 0x018

Reset value: 0x0000 0000

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

**CDCPRE[3:0]**

This field controls the divider ratio applied to the HSE oscillator output before dividing it for PLL1. The divider ratio is given as: 2^CDCCPRE[3:0] x PLLOUT. The divider ratio range is from 2 to 32768. Setting the CDCCPRE register to 0x00 represents the default setting of 1:

- **rw**: Read and write.
Caution: Care must be taken when using the voltage scaling. Due to the propagation delay of the new division factor, after a prescaler factor change and before lowering the VCORE voltage, this register must be read in order to check that the new prescaler value has been taken into account.

Depending on the clock source frequency and the voltage range, the software application must program a correct value in HPRE to make sure that the system frequency does not exceed the maximum frequency.
### 8.7.7 RCC CPU domain clock configuration register 2 (RCC_CDCFGR2)

Address offset: 0x01C  
Reset value: 0x0000 0000  

<table>
<thead>
<tr>
<th>Bit 31:11</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bits 10:8 | **CDPPRE2[2:0]**: CPU domain APB2 prescaler  
Set and reset by software to control the CPU domain APB2 clock division factor.  
The clock is divided by the new prescaler factor from 1 to 16 cycles of *rcc_hclk1* after CDPRE2 write.  
0xx: *rcc_pclk2 = rcc_hclk1* (default after reset)  
100: *rcc_pclk2 = rcc_hclk1 / 2*  
101: *rcc_pclk2 = rcc_hclk1 / 4*  
110: *rcc_pclk2 = rcc_hclk1 / 8*  
111: *rcc_pclk2 = rcc_hclk1 / 16*  
| Bit 7 | Reserved, must be kept at reset value. |
| Bits 6:4 | **CDPPRE1[2:0]**: CPU domain APB1 prescaler  
Set and reset by software to control the CPU domain APB1 clock division factor.  
The clock is divided by the new prescaler factor from 1 to 16 cycles of *rcc_hclk1* after CDPRE1 write.  
0xx: *rcc_pclk1 = rcc_hclk1* (default after reset)  
100: *rcc_pclk1 = rcc_hclk1 / 2*  
101: *rcc_pclk1 = rcc_hclk1 / 4*  
110: *rcc_pclk1 = rcc_hclk1 / 8*  
111: *rcc_pclk1 = rcc_hclk1 / 16*  
| Bits 3:0 | Reserved, must be kept at reset value. |

### 8.7.8 RCC SmartRun domain clock configuration register (RCC_SRDCFGR)

Address offset: 0x020  
Reset value: 0x0000 0000  

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bits 15:0 | **SRDPPRE[2:0]**: SmartRun domain prescaler  
Set and reset by software to control the SmartRun domain clock division factor.  
The clock is divided by the new prescaler factor from 1 to 16 cycles of *rcc_hclk1* after SRDPRE write.  
0xx: *rcc_pclk = rcc_hclk1* (default after reset)  
100: *rcc_pclk = rcc_hclk1 / 2*  
101: *rcc_pclk = rcc_hclk1 / 4*  
110: *rcc_pclk = rcc_hclk1 / 8*  
111: *rcc_pclk = rcc_hclk1 / 16*  
|
8.7.9 RCC PLLs clock source selection register (RCC_PLLCKSELR)

Address offset: 0x028
Reset value: 0x0202 0200

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:20 **DIVM3[5:0]:** prescaler for PLL3
   
   Set and cleared by software to configure the prescaler of the PLL3.
   
   The hardware does not allow any modification of this prescaler when PLL3 is enabled (PLL3ON = 1).
   
   In order to save power when PLL3 is not used, the value of DIVM3 must be set to 0.
   
   000000: prescaler disabled (default after reset)
   
   000001: division by 1 (bypass)
   
   000010: division by 2
   
   000011: division by 3
   
   ...
   
   100000: division by 32 (default after reset)
   
   ...
   
   111111: division by 63
   
Bits 19:18 Reserved, must be kept at reset value.

Bits 6:4 **SRDPPRE[2:0]:** SmartRun domain APB4 prescaler
   
   Set and reset by software to control the SmartRun domain APB4 clock division factor.
   
   The clock is divided by the new prescaler factor from 1 to 16 cycles of rcc_hclk4 after SRDPPRE write.
   
   0xx: rcc_pclk4 = rcc_hclk4 (default after reset)
   
   100: rcc_pclk4 = rcc_hclk4 / 2
   
   101: rcc_pclk4 = rcc_hclk4 / 4
   
   110: rcc_pclk4 = rcc_hclk4 / 8
   
   111: rcc_pclk4 = rcc_hclk4 / 16

Bits 3:0 Reserved, must be kept at reset value.
Bits 17:12  **DIVM2[5:0]:** prescaler for PLL2
- Set and cleared by software to configure the prescaler of the PLL2.
- The hardware does not allow any modification of this prescaler when PLL2 is enabled (PLL2ON = 1).
- In order to save power when PLL2 is not used, the value of DIVM2 must be set to 0.
  - 000000: prescaler disabled
  - 000001: division by 1 (bypass)
  - 000010: division by 2
  - 000011: division by 3
  ...  
  - 100000: division by 32 (default after reset)
  ...  
  - 111111: division by 63

Bits 11:10  Reserved, must be kept at reset value.

Bits 9:4  **DIVM1[5:0]:** prescaler for PLL1
- Set and cleared by software to configure the prescaler of the PLL1.
- The hardware does not allow any modification of this prescaler when PLL1 is enabled (PLL1ON = 1).
- In order to save power when PLL1 is not used, the value of DIVM1 must be set to 0.
  - 000000: prescaler disabled
  - 000001: division by 1 (bypass)
  - 000010: division by 2
  - 000011: division by 3
  ...  
  - 100000: division by 32 (default after reset)
  ...  
  - 111111: division by 63

Bits 3:2  Reserved, must be kept at reset value.

Bits 1:0  **PLLSRC[1:0]:** DIVMx and PLLs clock source selection
- Set and reset by software to select the PLL clock source.
- These bits can be written only when all PLLs are disabled.
- In order to save power, when no PLL is used, the value of PLLSRC must be set to ‘11’.
  - 00: HSI selected as PLL clock (hsi_ck) (default after reset)
  - 01: CSI selected as PLL clock (csi_ck)
  - 10: HSE selected as PLL clock (hse_ck)
  - 11: no clock send to DIVMx divider and PLLs
8.7.10 RCC PLLs configuration register (RCC_PLLCFGR)

Address offset: 0x02C  
Reset value: 0x01FF 0000

<table>
<thead>
<tr>
<th>Bit 31:25</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 24</td>
<td>DIVR3EN: PLL3 DIVR divider output enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software to enable the pll3_r_ck output of the PLL3.</td>
</tr>
<tr>
<td></td>
<td>To save power, DIVR3EN and DIVR3 bits must be set to 0 when the pll3_r_ck is not used.</td>
</tr>
<tr>
<td></td>
<td>This bit can be written only when the PLL3 is disabled (PLL3ON = 0 and PLL3RDY = 0).</td>
</tr>
<tr>
<td></td>
<td>0: pll3_r_ck output disabled</td>
</tr>
<tr>
<td></td>
<td>1: pll3_r_ck output enabled (default after reset)</td>
</tr>
<tr>
<td>Bit 23</td>
<td>DIVQ3EN: PLL3 DIVQ divider output enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software to enable the pll3_q_ck output of the PLL3.</td>
</tr>
<tr>
<td></td>
<td>To save power, DIVR3EN and DIVR3 bits must be set to 0 when the pll3_r_ck is not used.</td>
</tr>
<tr>
<td></td>
<td>This bit can be written only when the PLL3 is disabled (PLL3ON = 0 and PLL3RDY = 0).</td>
</tr>
<tr>
<td></td>
<td>0: pll3_q_ck output disabled</td>
</tr>
<tr>
<td></td>
<td>1: pll3_q_ck output enabled (default after reset)</td>
</tr>
<tr>
<td>Bit 22</td>
<td>DIVP3EN: PLL3 DIVP divider output enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software to enable the pll3_p_ck output of the PLL3.</td>
</tr>
<tr>
<td></td>
<td>This bit can be written only when the PLL3 is disabled (PLL3ON = 0 and PLL3RDY = 0).</td>
</tr>
<tr>
<td></td>
<td>To save power, DIVR3EN and DIVR3 bits must be set to 0 when the pll3_r_ck is not used.</td>
</tr>
<tr>
<td></td>
<td>0: pll3_p_ck output disabled</td>
</tr>
<tr>
<td></td>
<td>1: pll3_p_ck output enabled (default after reset)</td>
</tr>
<tr>
<td>Bit 21</td>
<td>DIVR2EN: PLL2 DIVR divider output enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software to enable the pll2_r_ck output of the PLL2.</td>
</tr>
<tr>
<td></td>
<td>To save power, DIVR2EN and DIVR2 bits must be set to 0 when the pll2_r_ck is not used.</td>
</tr>
<tr>
<td></td>
<td>This bit can be written only when the PLL2 is disabled (PLL2ON = 0 and PLL2RDY = 0).</td>
</tr>
<tr>
<td></td>
<td>0: pll2_r_ck output disabled</td>
</tr>
<tr>
<td></td>
<td>1: pll2_r_ck output enabled (default after reset)</td>
</tr>
<tr>
<td>Bit 20</td>
<td>DIVQ2EN: PLL2 DIVQ divider output enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software to enable the pll2_q_ck output of the PLL2.</td>
</tr>
<tr>
<td></td>
<td>To save power, DIVR2EN and DIVR2 bits must be set to 0 when the pll2_r_ck is not used.</td>
</tr>
<tr>
<td></td>
<td>This bit can be written only when the PLL2 is disabled (PLL2ON = 0 and PLL2RDY = 0).</td>
</tr>
<tr>
<td></td>
<td>0: pll2_q_ck output disabled</td>
</tr>
<tr>
<td></td>
<td>1: pll2_q_ck output enabled (default after reset)</td>
</tr>
</tbody>
</table>
Bit 19  **DIVP2EN**: PLL2 DIVP divider output enable
Set and reset by software to enable the `pll2_p_ck` output of the PLL2.
This bit can be written only when the PLL2 is disabled (PLL2ON = 0 and PLL2RDY = 0).
To save power, DIVR3EN and DIVR3 bits must be set to 0 when the `pll3_r_ck` is not used.
0: `pll2_p_ck` output disabled
1: `pll2_p_ck` output enabled (default after reset)

Bit 18  **DIVR1EN**: PLL1 DIVR divider output enable
Set and reset by software to enable the `pll1_r_ck` output of the PLL1.
To save power, DIVR3EN and DIVR3 bits must be set to 0 when the `pll3_r_ck` is not used.
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0: `pll1_r_ck` output disabled
1: `pll1_r_ck` output enabled (default after reset)

Bit 17  **DIVQ1EN**: PLL1 DIVQ divider output enable
Set and reset by software to enable the `pll1_q_ck` output of the PLL1.
In order to save power, when the `pll1_q_ck` output of the PLL1 is not used, the `pll1_q_ck` must be disabled.
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0: `pll1_q_ck` output disabled
1: `pll1_q_ck` output enabled (default after reset)

Bit 16  **DIVP1EN**: PLL1 DIVP divider output enable
Set and reset by software to enable the `pll1_p_ck` output of the PLL1.
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
In order to save power, when the `pll1_p_ck` output of the PLL1 is not used, the `pll1_p_ck` must be disabled.
0: `pll1_p_ck` output disabled
1: `pll1_p_ck` output enabled (default after reset)

Bits 15:12  Reserved, must be kept at reset value.

Bits 11:10  **PLL3RGE[1:0]**: PLL3 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL3.
These bits must be written before enabling the PLL3.
00: PLL3 input (`ref3_ck`) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL3 input (`ref3_ck`) clock range frequency between 2 and 4 MHz
10: PLL3 input (`ref3_ck`) clock range frequency between 4 and 8 MHz
11: PLL3 input (`ref3_ck`) clock range frequency between 8 and 16 MHz

Bit 9  **PLL3VCOSEL**: PLL3 VCO selection
Set and reset by software to select the proper VCO frequency range used for PLL3.
This bit must be written before enabling the PLL3.
0: wide VCO range 128 to 560 MHz (default after reset)
1: medium VCO range 150 to 420 MHz

Bit 8  **PLL3FRACEN**: PLL3 fractional latch enable
Set and reset by software to latch the content of FRACN3 into the sigma-delta modulator.
In order to latch the FRACN3 value into the sigma-delta modulator, PLL3FRACEN must be set to 0, then set to 1. The transition 0 to 1 transfers the content of FRACN3 into the modulator.
Refer to **PLL initialization phase on page 352** for additional information.
8.7.11 RCC PLL1 dividers configuration register (RCC_PLL1DIVR)

Address offset: 0x030
Reset value: 0x0101 0280

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PLL2RGE[1:0]</td>
<td>PLL2 input frequency range</td>
<td>PLL2 input (ref2_ck) clock range frequency between 1 and 2 MHz (default after reset)</td>
</tr>
<tr>
<td>1</td>
<td>PLL2VCOSEL</td>
<td>PLL2 VCO selection</td>
<td>PLL2 input (ref2_ck) clock range frequency between 2 and 4 MHz</td>
</tr>
<tr>
<td>2</td>
<td>PLL2FRACEN</td>
<td>PLL2 fractional latch enable</td>
<td>PLL2 input (ref2_ck) clock range frequency between 4 and 8 MHz</td>
</tr>
<tr>
<td>3</td>
<td>PLL1RGE[1:0]</td>
<td>PLL1 input frequency range</td>
<td>PLL1 input (ref1_ck) clock range frequency between 1 and 2 MHz (default after reset)</td>
</tr>
<tr>
<td>4</td>
<td>PLL1VCOSEL</td>
<td>PLL1 VCO selection</td>
<td>PLL1 input (ref1_ck) clock range frequency between 2 and 4 MHz</td>
</tr>
<tr>
<td>5</td>
<td>PLL1FRACEN</td>
<td>PLL1 fractional latch enable</td>
<td>PLL1 input (ref1_ck) clock range frequency between 4 and 8 MHz</td>
</tr>
</tbody>
</table>

**Register Description**

**RCC_PLL1DIVR**

- **Address offset:** 0x030
- **Reset value:** 0x0101 0280

**Bits 7:6 PLL2RGE[1:0]**
- PLL2 input frequency range
  - PLL2 input (ref2_ck) clock range frequency between 1 and 2 MHz (default after reset)
  - PLL2 input (ref2_ck) clock range frequency between 2 and 4 MHz
  - PLL2 input (ref2_ck) clock range frequency between 4 and 8 MHz
  - PLL2 input (ref2_ck) clock range frequency between 8 and 16 MHz

**Bit 5 PLL2VCOSEL**
- PLL2 VCO selection
  - PLL2 input (ref2_ck) clock range frequency between 128 to 560 MHz (default after reset)
  - PLL2 input (ref2_ck) clock range frequency between 150 to 420 MHz

**Bit 4 PLL2FRACEN**
- PLL2 fractional latch enable
  - PLL2 input (ref2_ck) clock range frequency between 128 to 560 MHz (default after reset)

**Bits 3:2 PLL1RGE[1:0]**
- PLL1 input frequency range
  - PLL1 input (ref1_ck) clock range frequency between 1 and 2 MHz (default after reset)
  - PLL1 input (ref1_ck) clock range frequency between 2 and 4 MHz
  - PLL1 input (ref1_ck) clock range frequency between 4 and 8 MHz

**Bit 1 PLL1VCOSEL**
- PLL1 VCO selection
  - PLL1 input (ref1_ck) clock range frequency between 128 to 560 MHz (default after reset)
  - PLL1 input (ref1_ck) clock range frequency between 150 to 420 MHz

**Bit 0 PLL1FRACEN**
- PLL1 fractional latch enable
  - PLL1 input (ref1_ck) clock range frequency between 128 to 560 MHz (default after reset)

Refer to *PLL initialization phase on page 352* for additional information.
Bit 31  Reserved, must be kept at reset value.

Bits 30:24  **DIVR1[6:0]**: PLL1 DIVR division factor

Set and reset by software to control the frequency of the **pll1_r_ck** clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).

- 0000000:  \( \text{pll1_r_ck} = vco1_ck \)
- 0000001:  \( \text{pll1_r_ck} = vco1_ck / 2 \) (default after reset)
- 0000010:  \( \text{pll1_r_ck} = vco1_ck / 3 \)
- 0000011:  \( \text{pll1_r_ck} = vco1_ck / 4 \)
  ...
- 1111111:  \( \text{pll1_r_ck} = vco1_ck / 128 \)

Bit 23  Reserved, must be kept at reset value.

Bits 22:16  **DIVQ1[6:0]**: PLL1 DIVQ division factor

Set and reset by software to control the frequency of the **pll1_q_ck** clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).

- 0000000:  \( \text{pll1_q_ck} = vco1_ck \)
- 0000001:  \( \text{pll1_q_ck} = vco1_ck / 2 \) (default after reset)
- 0000010:  \( \text{pll1_q_ck} = vco1_ck / 3 \)
- 0000011:  \( \text{pll1_q_ck} = vco1_ck / 4 \)
  ...
- 1111111:  \( \text{pll1_q_ck} = vco1_ck / 128 \)

Bits 15:9  **DIVP1[6:0]**: PLL1 DIVP division factor

Set and reset by software to control the frequency of the **pll1_p_ck** clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).

Note that odd division factors are not allowed.

- 0000000:  not allowed
- 0000001:  \( \text{pll1_p_ck} = vco1_ck / 2 \) (default after reset)
- 0000010:  not allowed
- 0000011:  \( \text{pll1_p_ck} = vco1_ck / 4 \)
  ...
- 1111111:  \( \text{pll1_p_ck} = vco1_ck / 128 \)

Bits 8:0  **DIVN1[8:0]**: multiplication factor for PLL1 VCO

Set and reset by software to control the multiplication factor of the VCO.
These bits can be written only when the PLL1 is disabled (PLL1ON = PLL1RDY = 0).

- ..........:  not used
- 0x006:  wrong configuration
- 0x007:  \( \text{DIVN1} = 8 \)
  ...
- 0x080:  \( \text{DIVN1} = 129 \) (default after reset)
  ...
- 0x1A3:  \( \text{DIVN1} = 420 \)

Others:  wrong configurations

**Caution:**  The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:

- 128 to 560 MHz if PLL1VCOSEL = 0
- 150 to 420 MHz if PLL1VCOSEL = 1

VCO output frequency = \( F_{\text{ref1_ck}} \times \text{DIVN1} \), when fractional value 0 has been loaded into FRACN1, with:

- \( \text{DIVN1} \) between 8 and 420
- The input frequency \( F_{\text{ref1_ck}} \) must be between 1 and 16 MHz.
8.7.12  RCC PLL1 fractional divider register (RCC_PLL1FRACR)

Address offset: 0x034
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Bit 23:16</th>
<th>Bit 15:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

FRACN[12:0]

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 FRACN[12:0]: fractional part of the multiplication factor for PLL1 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO. These bits can be written at any time, allowing dynamic fine-tuning of the PLL1 VCO.

Caution: The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:

- 128 to 560 MHz if PLL1VCOSEL = 0
- 150 to 420 MHz if PLL1VCOSEL = 1

VCO output frequency = \( F_{\text{ref1,ck}} \times (\text{DIVN1} + (\text{FRACN1} / 2^{13})) \), with

- DIVN1 between 8 and 420
- FRACN1 can be between 0 and \( 2^{13} - 1 \)
- The input frequency \( F_{\text{ref1,ck}} \) must be between 1 and 16 MHz.

To change the FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

- Set the bit PLL1FRACEN to 0.
- Write the new fractional value into FRACN1.
- Set the bit PLL1FRACEN to 1.

Bits 2:0 Reserved, must be kept at reset value.

8.7.13  RCC PLL2 dividers configuration register (RCC_PLL2DIVR)

Address offset: 0x038
Reset value: 0x0101 0280

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Bit 23:16</th>
<th>Bit 15:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>DIVR2[6:0]</td>
<td>Reserved</td>
<td>DIVQ2[6:0]</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Bit 23:16</th>
<th>Bit 15:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIVP2[6:0]</td>
<td>DIVN2[8:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

398/2974  RM0455 Rev 3
Bit 31  Reserved, must be kept at reset value.

Bits 30:24  DIVR2[6:0]: PLL2 DIVR division factor
Set and reset by software to control the frequency of the pll2_r_ck clock.
These bits can be written only when the PLL2 is disabled (PLL2ON = PLL2RDY = 0).
0000000: pll2_r_ck = vco2_ck
0000001: pll2_r_ck = vco2_ck / 2 (default after reset)
0000010: pll2_r_ck = vco2_ck / 3
0000011: pll2_r_ck = vco2_ck / 4
...
1111111: pll2_r_ck = vco2_ck / 128

Bit 23  Reserved, must be kept at reset value.

Bits 22:16  DIVQ2[6:0]: PLL2 DIVQ division factor
Set and reset by software to control the frequency of the pll2_q_ck clock.
These bits can be written only when the PLL2 is disabled (PLL2ON = PLL2RDY = 0).
0000000: pll2_q_ck = vco2_ck
0000001: pll2_q_ck = vco2_ck / 2 (default after reset)
0000010: pll2_q_ck = vco2_ck / 3
0000011: pll2_q_ck = vco2_ck / 4
...
1111111: pll2_q_ck = vco2_ck / 128

Bits 15:9  DIVP2[6:0]: PLL2 DIVP division factor
Set and reset by software to control the frequency of the pll2_p_ck clock.
These bits can be written only when the PLL2 is disabled (PLL2ON = PLL2RDY = 0).
0000000: pll2_p_ck = vco2_ck
0000001: pll2_p_ck = vco2_ck / 2 (default after reset)
0000010: pll2_p_ck = vco2_ck / 3
0000011: pll2_p_ck = vco2_ck / 4
...
1111111: pll2_p_ck = vco2_ck / 128

Bits 8:0  DIVN2[8:0]: multiplication factor for PLL2 VCO
Set and reset by software to control the multiplication factor of the VCO.
These bits can be written only when the PLL is disabled (PLL2ON = PLL2RDY = 0).
.......: not used
0x006: wrong configuration
0x007: DIVN2 = 8
...
0x080: DIVN2 = 129 (default after reset)
...
0x1A3: DIVN2 = 420
Others: wrong configurations

Caution:  The software must set correctly these bits to insure that the VCO output frequency is
between its valid frequency range, that is:
– 128 to 560 MHz if PLL2VCOSEL = 0
– 150 to 420 MHz if PLL2VCOSEL = 1
VCO output frequency = F_{ref2_ck} x DIVN2, when fractional value 0 has been loaded into FRACN2,
with
– DIVN2 between 8 and 420
– The input frequency F_{ref2_ck} must be between 1 and 16MHz.
8.7.14 RCC PLL2 fractional divider register (RCC_PLL2FRACR)

Address offset: 0x03C
Reset value: 0x0000 0000

| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:3  | FRACN2[12:0]: fractional part of the multiplication factor for PLL2 VCO |
|           | Set and reset by software to control the fractional part of the multiplication factor of the VCO. |
|           | These bits can be written at any time, allowing dynamic fine-tuning of the PLL2 VCO. |
| **Caution:** | The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is: |
|           | – 128 to 560 MHz if PLL2VCOSEL = 0 |
|           | – 150 to 420 MHz if PLL2VCOSEL = 1 |
|           | VCO output frequency = Fref2_ck x (DIVN2 + (FRACN2 / 213)), with |
|           | – DIVN2 between 8 and 420 |
|           | – FRACN2 can be between 0 and 213 - 1 |
|           | – The input frequency Fref2_ck must be between 1 and 16 MHz. |
|           | In order to change the FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows: |
|           | – Set the bit PLL2FRACEN to 0. |
|           | – Write the new fractional value into FRACN2. |
|           | – Set the bit PLL2FRACEN to 1. |
| Bits 2:0  | Reserved, must be kept at reset value. |

8.7.15 RCC PLL3 dividers configuration register (RCC_PLL3DIVR)

Address offset: 0x040
Reset value: 0x0101 0280

| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 25:0  | Reserved, must be kept at reset value. |

---

400/2974 RM0455 Rev 3
Bit 31: Reserved, must be kept at reset value.

**Bits 30:24**  
DIVR3[6:0]: PLL3 DIVR division factor  
Set and reset by software to control the frequency of the \( \text{pll3}_r\_ck \) clock.  
These bits can be written only when the PLL3 is disabled (PLL3ON = PLL3RDY = 0).  
- 0000000: \( \text{pll3}_r\_ck = \text{vco3}_c\_k \)  
- 0000001: \( \text{pll3}_r\_ck = \text{vco3}_c\_k / 2 \) (default after reset)  
- 0000010: \( \text{pll3}_r\_ck = \text{vco3}_c\_k / 3 \)  
- 0000011: \( \text{pll3}_r\_ck = \text{vco3}_c\_k / 4 \)  
...  
- 1111111: \( \text{pll3}_r\_ck = \text{vco3}_c\_k / 128 \)

Bit 23: Reserved, must be kept at reset value.

**Bits 22:16**  
DIVQ3[6:0]: PLL3 DIVQ division factor  
Set and reset by software to control the frequency of the \( \text{pll3}_q\_ck \) clock.  
These bits can be written only when the PLL3 is disabled (PLL3ON = PLL3RDY = 0).  
- 0000000: \( \text{pll3}_q\_ck = \text{vco3}_c\_k \)  
- 0000001: \( \text{pll3}_q\_ck = \text{vco3}_c\_k / 2 \) (default after reset)  
- 0000010: \( \text{pll3}_q\_ck = \text{vco3}_c\_k / 3 \)  
- 0000011: \( \text{pll3}_q\_ck = \text{vco3}_c\_k / 4 \)  
...  
- 1111111: \( \text{pll3}_q\_ck = \text{vco3}_c\_k / 128 \)

**Bits 15:9**  
DIVP3[6:0]: PLL3 DIVP division factor  
Set and reset by software to control the frequency of the \( \text{pll3}_p\_ck \) clock.  
These bits can be written only when the PLL3 is disabled (PLL3ON = PLL3RDY = 0).  
- 0000000: \( \text{pll3}_p\_ck = \text{vco3}_c\_k \)  
- 0000001: \( \text{pll3}_p\_ck = \text{vco3}_c\_k / 2 \) (default after reset)  
- 0000010: \( \text{pll3}_p\_ck = \text{vco3}_c\_k / 3 \)  
- 0000011: \( \text{pll3}_p\_ck = \text{vco3}_c\_k / 4 \)  
...  
- 1111111: \( \text{pll3}_p\_ck = \text{vco3}_c\_k / 128 \)

**Bits 8:0**  
DIVN3[8:0]: Multiplication factor for PLL3 VCO  
Set and reset by software to control the multiplication factor of the VCO.  
These bits can be written only when the PLL is disabled (PLL3ON = PLL3RDY = 0).  
- 0x006: wrong configuration  
- 0x007: DIVN3 = 8  
...  
- 0x080: DIVN3 = 129 (default after reset)  
...  
- 0x1A3: DIVN3 = 420  
Others: wrong configurations

**Caution:** The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:  
- 128 to 560 MHz if PLL3VCORESEL = 0  
- 150 to 420 MHz if PLL3VCORESEL = 1  
VCO output frequency = \( F_{\text{ref3}_c\_k} \times \text{DIVN3} \), when fractional value 0 has been loaded into FRACN3, with:  
- \( \text{DIVN3} \) between 8 and 420  
- The input frequency \( F_{\text{ref3}_c\_k} \) must be between 1 and 16MHz
8.7.16 RCC PLL3 fractional divider register (RCC_PLL3FRACR)

Address offset: 0x044
Reset value: 0x0000 0000

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 FRACN3[12:0]: fractional part of the multiplication factor for PLL3 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO. These bits can be written at any time, allowing dynamic fine-tuning of the PLL3 VCO.

**Caution:** The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:

- 128 to 560 MHz if PLL3VCOSEL = 0
- 150 to 420 MHz if PLL3VCOSEL = 1

VCO output frequency = \( F_{\text{ref3,ck}} \times (\text{DIVN3} + (\text{FRACN3} / 2^{13})) \), with

- DIVN3 between 8 and 420
- FRACN3 can be between 0 and \( 2^{13} - 1 \)
- The input frequency \( F_{\text{ref3,ck}} \) must be between 1 and 16 MHz.

In order to change the FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

- Set the bit PLL1FRACEN to 0.
- Write the new fractional value into FRACN1.
- Set the bit PLL1FRACEN to 1.

Bits 2:0 Reserved, must be kept at reset value.

8.7.17 RCC CPU domain kernel clock configuration register (RCC_CDCCIPR)

Address offset: 0x04C
Reset value: 0x0000 0000

Changing the clock source on-the-fly is allowed and does not generate any timing violation. However the user must make sure that both the previous and the new clock sources are present during the switching, and during the whole transition time. Refer to *Clock switches and gating.*
8.7.18 RCC CPU domain kernel clock configuration register (RCC_CDCCIP1R)

Address offset: 0x050

Reset value: 0x0000 0000

Changing the clock source on-the-fly is allowed and does not generate any timing violation. However the user must make sure that both the previous and the new clock sources are present during the switching, and during the whole transition time. Refer to Clock switches and gating.
Bit 31  **SWPSEL**: SWP kernel clock source selection
- Set and reset by software.
  - 0: **pclk** selected as SWP kernel clock (default after reset)
  - 1: **hsi_ker_ck** selected as SWP kernel clock

Bit 30  Reserved, must be kept at reset value.

Bits 29:28  **FDCANSEL[1:0]**: FDCAN kernel clock source selection
- Set and reset by software.
  - 00: **hse_ck** clock selected as FDCAN kernel clock (default after reset)
  - 01: **pll1_q_ck** selected as FDCAN kernel clock
  - 10: **pll2_q_ck** selected as FDCAN kernel clock
  - 11: reserved, the kernel clock is disabled

Bits 27:25  Reserved, must be kept at reset value.

Bit 24  **DFSDM1SEL**: DFSDM1 kernel clock Clk source selection
- Set and reset by software.
  - Note: the DFSDM1 Aclk clock source selection is done by SAI1SEL (see Figure 53).
  - 0: **rcc_pclk2** selected as DFSDM1 Clk kernel clock (default after reset)
  - 1: **sys_ck** selected as DFSDM1 Clk kernel clock

Bits 23:22  Reserved, must be kept at reset value.

Bits 21:20  **SPDIFRXSEL[1:0]**: SPDIFRX kernel clock source selection
  - 00: **pll1_q_ck** selected as SPDIFRX kernel clock (default after reset)
  - 01: **pll2_r_ck** selected as SPDIFRX kernel clock
  - 10: **pll3_r_ck** selected as SPDIFRX kernel clock
  - 11: **hsi_ker_ck** selected as SPDIFRX kernel clock

Bit 19  Reserved, must be kept at reset value.

Bits 18:16  **SPI45SEL[2:0]**: SPI4 and 5 kernel clock source selection
- Set and reset by software.
  - 000: APB clock selected as kernel clock (default after reset)
  - 001: **pll2_q_ck** is selected as kernel clock
  - 010: **pll3_q_ck** is selected as kernel clock
  - 011: **hsi_ker_ck** is selected as kernel clock
  - 100: **csi_ker_ck** is selected as kernel clock
  - 101: **hse_ck** is selected as kernel clock
  - others: reserved, the kernel clock is disabled
Bit 15 Reserved, must be kept at reset value.

Bits 14:12  **SPI123SEL[2:0]**: SPI/I2S1,2 and 3 kernel clock source selection  
Set and reset by software.

**Caution:** If the selected clock is the external clock and this clock is stopped, it is not be possible to switch to another clock. Refer to *Clock switches and gating on page 373* for additional information.

- 000: **pll1_q_ck** selected as SPI/I2S1,2 and 3 kernel clock (default after reset)
- 001: **pll2_p_ck** selected as SPI/I2S1,2 and 3 kernel clock
- 010: **pll3_p_ck** selected as SPI/I2S1,2 and 3 kernel clock
- 011: **I2S_CKIN** selected as SPI/I2S1,2 and 3 kernel clock  
- 100: **per_ck** selected as SPI/I2S1,2 and 3 kernel clock  
- others: reserved, the kernel clock is disabled  

Note: **I2S_CKIN** is an external clock taken from a pin.

Bits 11:9  **SAI2BSEL[2:0]**: SAI2 kernel clock B source selection  
Set and reset by software.

**Caution:** If the selected clock is the external clock and this clock is stopped, it is not be possible to switch to another clock. Refer to *Clock switches and gating on page 373* for additional information.

- 000: **pll1_q_ck** selected as SAI2 kernel clock B (default after reset)
- 001: **pll2_p_ck** selected as SAI2 kernel clock B
- 010: **pll3_p_ck** selected as SAI2 kernel clock B
- 011: **I2S_CKIN** selected as SAI2 kernel clock B  
- 100: **per_ck** selected as SAI2 kernel clock B
- 101: **spdifrx_symb_ck** selected as SAI2 kernel clock B  
- others: reserved, the kernel clock is disabled  

Note: **I2S_CKIN** is an external clock taken from a pin. **spdifrx_symb_ck** is the symbol clock generated by the spdifrx (see Figure 53).
8.7.19 RCC CPU domain kernel clock configuration register
(RCC_CDCCIP2R)

Address offset: 0x054

Reset value: 0x0000 0000

Changing the clock source on-the-fly is allowed and does not generate any timing violation. However the user must make sure that both the previous and the new clock sources are present during the switching, and during the whole transition time. Refer to Clock switches and gating.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM1SEL[2:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 15</td>
<td>Bit 14</td>
<td>Bit 13</td>
<td>Bit 12</td>
<td>Bit 11</td>
<td>Bit 10</td>
<td>Bit 9</td>
<td>Bit 8</td>
<td>Bit 7</td>
<td>Bit 6</td>
<td>Bit 5</td>
<td>Bit 4</td>
<td>Bit 3</td>
<td>Bit 2</td>
<td>Bit 1</td>
<td>Bit 0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Caution: If the selected clock is the external clock and this clock is stopped, it is not possible to switch to another clock. Refer to Clock switches and gating on page 373 for additional information.

Note: I2S_CKIN is an external clock taken from a pin. spdifrx_symb_ck is the symbol clock generated by the spdifrx (see Figure 53).
Bit 31  Reserved, must be kept at reset value.

Bits 30:28  **LPTIM1SEL[2:0]**: LPTIM1 kernel clock source selection
Set and reset by software.
000: **rcc_pclk1** selected as kernel peripheral clock (default after reset)
001: **pll2_p_ck** selected as kernel peripheral clock
010: **pll3_r_ck** selected as kernel peripheral clock
011: **lse_ck** selected as kernel peripheral clock
100: **lsi_ck** selected as kernel peripheral clock
101: **per_ck** selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

Bits 27:24  Reserved, must be kept at reset value.

Bits 23:22  **CECSEL[1:0]**: HDMI-CEC kernel clock source selection
Set and reset by software.
00: **lse_ck** selected as kernel clock (default after reset)
01: **lsi_ck** selected as kernel clock
10: **csi_ker_ck** divided by 122 selected as kernel clock
11: reserved, the kernel clock is disabled

Bits 21:20  **USBSEL[1:0]**: USBOTG 1 and 2 kernel clock source selection
Set and reset by software.
00: Disable the kernel clock (default after reset)
01: **pll1_q_ck** selected as kernel clock
10: **pll3_q_ck** selected as kernel clock
11: **hsi48_ck** selected as kernel clock

Bits 19:14  Reserved, must be kept at reset value.

Bits 13:12  **I2C123SEL[1:0]**: I2C1,2,3 kernel clock source selection
Set and reset by software.
00: **rcc_pclk1** selected as kernel clock (default after reset)
01: **pll3_r_ck** selected as kernel clock
10: **hsi_ker_ck** selected as kernel clock
11: **csi_ker_ck** selected as kernel clock

Bits 11:10  Reserved, must be kept at reset value.

Bits 9:8  **RNGSEL[1:0]**: RNG kernel clock source selection
Set and reset by software.
00: **hsi48_ck** selected as kernel clock (default after reset)
01: **pll1_q_ck** selected as kernel clock
10: **lse_ck** selected as kernel clock
11: **lsi_ck** selected as kernel clock
8.7.20 RCC SmartRun domain kernel clock configuration register
(RCC_SRDCCIPR)

Address offset: 0x058

Reset value: 0x0000 0000

Changing the clock source on-the-fly is allowed and does not generate any timing violation. However the user must make sure that both the previous and the new clock sources are present during the switching, and during the whole transition time. Refer to Clock switches and gating.

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

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

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:3 **USART16910SEL[2:0]**: USART1, 6, 9 and 10 kernel clock source selection

Set and reset by software.

000: **rcc_pclk2** selected as kernel clock (default after reset)

001: **pll2_q_ck** selected as kernel clock

010: **pll3_q_ck** selected as kernel clock

011: **hsi_ker_ck** selected as kernel clock

100: **csi_ker_ck** selected as kernel clock

101: **lse_ck** selected as kernel clock

others: reserved, the kernel clock is disabled

Bits 2:0 **USART234578SEL[2:0]**: USART2/3, UART4,5, 7 and 8 (APB1) kernel clock source selection

Set and reset by software.

000: **rcc_pclk1** selected as kernel clock (default after reset)

001: **pll2_q_ck** selected as kernel clock

010: **pll3_q_ck** selected as kernel clock

011: **hsi_ker_ck** selected as kernel clock

100: **csi_ker_ck** selected as kernel clock

101: **lse_ck** selected as kernel clock

others: reserved, the kernel clock is disabled
Bit 31  Reserved, must be kept at reset value.

Bits 30:28  SPI6SEL[2:0]: SPI6 kernel clock source selection
Set and reset by software.
000: rcc_pclk4 selected as kernel peripheral clock (default after reset)
001: pll2_q_ck selected as kernel peripheral clock
010: pll3_q_ck selected as kernel peripheral clock
011: hsi_ker_ck selected as kernel peripheral clock
100: csi_ker_ck selected as kernel peripheral clock
101: hse_ck selected as kernel peripheral clock
110: I2S_CKIN selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

Bit 27  DFSDM2SEL: DFSDM2 kernel Clk clock source selection
Set and reset by software.
Note: The DFSDM2 Aclk clock source selection is done by SPI6SEL (see Figure 53 and Figure 54).
0: rcc_pclk4 selected as DFSDM2 Clk kernel clock (default after reset)
1: sys_ck selected as DFSDM2 Clk kernel clock

Bits 26:18  Reserved, must be kept at reset value.

Bits 17:16  ADCSEL[1:0]: SAR ADC kernel clock source selection
Set and reset by software.
00: pll2_p_ck selected as kernel peripheral clock (default after reset)
01: pll3_r_ck selected as kernel peripheral clock
10: per_ck selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

Bits 15:13  LPTIM3SEL[2:0]: LPTIM3 kernel clock source selection
Set and reset by software.
000: rcc_pclk4 selected as kernel peripheral clock (default after reset)
001: pll2_p_ck selected as kernel peripheral clock
010: pll3_r_ck selected as kernel peripheral clock
011: lse_ck selected as kernel peripheral clock
100: lsi_ck selected as kernel peripheral clock
101: per_ck selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

Bits 12:10  LPTIM2SEL[2:0]: LPTIM2 kernel clock source selection
Set and reset by software.
000: rcc_pclk4 selected as kernel peripheral clock (default after reset)
001: pll2_p_ck selected as kernel peripheral clock
010: pll3_r_ck selected as kernel peripheral clock
011: lse_ck selected as kernel peripheral clock
100: lsi_ck selected as kernel peripheral clock
101: per_ck selected as kernel peripheral clock
others: reserved, the kernel clock is disabled
8.7.21 RCC clock source interrupt enable register (RCC_CIER)

Address offset: 0x060

Reset value: 0x0000 0000

Bits 9:8  I2C4SEL[1:0]: I2C4 kernel clock source selection
Set and reset by software.
00: rcc_pclk4 selected as kernel peripheral clock (default after reset)
01: pll3_r_ck selected as kernel peripheral clock
10: hsi_ker_ck selected as kernel peripheral clock
11: csi_ker_ck selected as kernel peripheral clock

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

Bits 2:0  LPUART1SEL[2:0]: LPUART1 kernel clock source selection
Set and reset by software.
000: rcc_pclk4 selected as kernel peripheral clock (default after reset)
001: pll2_q_ck selected as kernel peripheral clock
010: pll3_q_ck selected as kernel peripheral clock
011: hsi_ker_ck selected as kernel peripheral clock
100: csi_ker_ck selected as kernel peripheral clock
101: lse_ck selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

Bit 9  LSECSSIE: LSE clock security system interrupt enable
Set and reset by software to enable/disable interrupt caused by the clock security system (CSS) on external 32 kHz oscillator.
0: LSE CSS interrupt disabled (default after reset)
1: LSE CSS interrupt enabled

Bit 8  PLL3RDYIE: PLL3 ready interrupt enable
Set and reset by software to enable/disable interrupt caused by PLL3 lock.
0: PLL3 lock interrupt disabled (default after reset)
1: PLL3 lock interrupt enabled

Bit 7  PLL2RDYIE: PLL2 ready interrupt enable
Set and reset by software to enable/disable interrupt caused by PLL2 lock.
0: PLL2 lock interrupt disabled (default after reset)
1: PLL2 lock interrupt enabled
**Bit 6**  **PLL1RDYIE**: PLL1 ready interrupt enable  
Set and reset by software to enable/disable interrupt caused by PLL1 lock.  
0: PLL1 lock interrupt disabled (default after reset)  
1: PLL1 lock interrupt enabled

**Bit 5**  **HSI48RDYIE**: HSI48 ready interrupt enable  
Set and reset by software to enable/disable interrupt caused by the HSI48 oscillator stabilization.  
0: HSI48 ready interrupt disabled (default after reset)  
1: HSI48 ready interrupt enabled

**Bit 4**  **CSIRDYIE**: CSI ready interrupt enable  
Set and reset by software to enable/disable interrupt caused by the CSI oscillator stabilization.  
0: CSI ready interrupt disabled (default after reset)  
1: CSI ready interrupt enabled

**Bit 3**  **HSERDYIE**: HSE ready interrupt enable  
Set and reset by software to enable/disable interrupt caused by the HSE oscillator stabilization.  
0: HSE ready interrupt disabled (default after reset)  
1: HSE ready interrupt enabled

**Bit 2**  **HSIRDYIE**: HSI ready interrupt enable  
Set and reset by software to enable/disable interrupt caused by the HSI oscillator stabilization.  
0: HSI ready interrupt disabled (default after reset)  
1: HSI ready interrupt enabled

**Bit 1**  **LSERDYIE**: LSE ready interrupt enable  
Set and reset by software to enable/disable interrupt caused by the LSE oscillator stabilization.  
0: LSE ready interrupt disabled (default after reset)  
1: LSE ready interrupt enabled

**Bit 0**  **LSIRDYIE**: LSI ready interrupt enable  
Set and reset by software to enable/disable interrupt caused by the LSI oscillator stabilization.  
0: LSI ready interrupt disabled (default after reset)  
1: LSI ready interrupt enabled

### 8.7.22  **RCC clock source interrupt flag register (RCC_CIFR)**

Address offset: 0x64  
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:11  Reserved, must be kept at reset value.

Bit 10  **HSECSSF**: HSE clock security system interrupt flag
        Reset by software by writing HSECSSC bit.
        Set by hardware in case of HSE clock failure.
        0: no clock security interrupt caused by HSE clock failure (default after reset)
        1: clock security interrupt caused by HSE clock failure

Bit 9  **LSECSSF**: LSE clock security system interrupt flag
        Reset by software by writing LSECSSC bit.
        Set by hardware when a failure is detected on the external 32 kHz oscillator and LSECSSIE is set.
        0: no failure detected on the external 32 kHz oscillator (default after reset)
        1: failure detected on the external 32 kHz oscillator

Bit 8  **PLL3RDYF**: PLL3 ready interrupt flag
        Reset by software by writing PLL3RDYC bit.
        Set by hardware when the PLL3 locks and PLL3RDYIE is set.
        0: no clock ready interrupt caused by PLL3 lock (default after reset)
        1: clock ready interrupt caused by PLL3 lock

Bit 7  **PLL2RDYF**: PLL2 ready interrupt flag
        Reset by software by writing PLL2RDYC bit.
        Set by hardware when the PLL2 locks and PLL2RDYIE is set.
        0: no clock ready interrupt caused by PLL2 lock (default after reset)
        1: clock ready interrupt caused by PLL2 lock

Bit 6  **PLL1RDYF**: PLL1 ready interrupt flag
        Reset by software by writing PLL1RDYC bit.
        Set by hardware when the PLL1 locks and PLL1RDYIE is set.
        0: no clock ready interrupt caused by PLL1 lock (default after reset)
        1: clock ready interrupt caused by PLL1 lock

Bit 5  **HSI48RDYF**: HSI48 ready interrupt flag
        Reset by software by writing HSI48RDYC bit.
        Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set.
        0: no clock ready interrupt caused by the HSI48 oscillator (default after reset)
        1: clock ready interrupt caused by the HSI48 oscillator

Bit 4  **CSIRDYF**: CSI ready interrupt flag
        Reset by software by writing CSIRDYC bit.
        Set by hardware when the CSI clock becomes stable and CSIRDYIE is set.
        0: no clock ready interrupt caused by the CSI (default after reset)
        1: clock ready interrupt caused by the CSI

Bit 3  **HSERDYF**: HSE ready interrupt flag
        Reset by software by writing HSERDYC bit.
        Set by hardware when the HSE clock becomes stable and HSERDYIE is set.
        0: no clock ready interrupt caused by the HSE (default after reset)
        1: clock ready interrupt caused by the HSE
8.7.23  RCC clock source interrupt clear register (RCC_CICR)

Address offset: 0x68
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.

Bit 10  HSECSSC: HSE clock security system interrupt clear
Set by software to clear HSECSSF.
Reset by hardware when clear done.
0: HSECSSF no effect (default after reset)
1: HSECSSF cleared

Bit 9  LSECSSC: LSE clock security system interrupt clear
Set by software to clear LSECSSF.
Reset by hardware when clear done.
0: LSECSSF no effect (default after reset)
1: LSECSSF cleared

Bit 8  PLL3RDYC: PLL3 ready interrupt clear
Set by software to clear PLL3RDYF.
Reset by hardware when clear done.
0: PLL3RDYF no effect (default after reset)
1: PLL3RDYF cleared

Bit 2  HSIRDYF: HSI ready interrupt flag
Reset by software by writing HSIRDYC bit.
Set by hardware when the HSI clock becomes stable and HSIRDYIE is set.
0: no clock ready interrupt caused by the HSI (default after reset)
1: clock ready interrupt caused by the HSI

Bit 1  LSIRDYF: LSE ready interrupt flag
Reset by software by writing LSIRDYC bit.
Set by hardware when the LSE clock becomes stable and LSERDYIE is set.
0: no clock ready interrupt caused by the LSE (default after reset)
1: clock ready interrupt caused by the LSE

Bit 0  LSIRDYF: LSI ready interrupt flag
Reset by software by writing LSIRDYC bit.
Set by hardware when the LSI clock becomes stable and LSIIRDYIE is set.
0: no clock ready interrupt caused by the LSI (default after reset)
1: clock ready interrupt caused by the LSI
Bit 7 **PLL2RDYC**: PLL2 ready interrupt clear  
Set by software to clear PLL2RDYF.  
Reset by hardware when clear done.  
0: PLL2RDYF no effect (default after reset)  
1: PLL2RDYF cleared  

Bit 6 **PLL1RDYC**: PLL1 ready interrupt clear  
Set by software to clear PLL1RDYF.  
Reset by hardware when clear done.  
0: PLL1RDYF no effect (default after reset)  
1: PLL1RDYF cleared  

Bit 5 **HSI48RDYC**: HSI48 ready interrupt clear  
Set by software to clear HSI48RDYF.  
Reset by hardware when clear done.  
0: HSI48RDYF no effect (default after reset)  
1: HSI48RDYF cleared  

Bit 4 **CSIRDYC**: CSI ready interrupt clear  
Set by software to clear CSIRDYF.  
Reset by hardware when clear done.  
0: CSIRDYF no effect (default after reset)  
1: CSIRDYF cleared  

Bit 3 **HSERDYC**: HSE ready interrupt clear  
Set by software to clear HSERDYF.  
Reset by hardware when clear done.  
0: HSERDYF no effect (default after reset)  
1: HSERDYF cleared  

Bit 2 **HSIRDYC**: HSI ready interrupt clear  
Set by software to clear HSIRDYF.  
Reset by hardware when clear done.  
0: HSIRDYF no effect (default after reset)  
1: HSIRDYF cleared  

Bit 1 **LSERDYC**: LSE ready interrupt clear  
Set by software to clear LSERDYF.  
Reset by hardware when clear done.  
0: LSERDYF no effect (default after reset)  
1: LSERDYF cleared  

Bit 0 **LSIRDYC**: LSI ready interrupt clear  
Set by software to clear LSIRDYF.  
Reset by hardware when clear done.  
0: LSIRDYF no effect (default after reset)  
1: LSIRDYF cleared
8.7.24 RCC Backup domain control register (RCC_BDCR)

Address offset: 0x070
Reset value: 0x0000 0000

Reset by Backup domain reset.

Access: 0 ≤ wait state ≤ 7, word, half-word and byte access. Wait states are inserted in case of successive accesses to this register.

After a system reset, the RCC_BDCR register is write-protected. To modify this register, the DBP bit in the PWR control register 1 (PWR_CR1) must be set to 1. RCC_BDCR bits are only reset after a Backup domain reset (see Section 8.4.6: Backup domain reset). Any other internal or external reset does not have any effect on these bits.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
| 16  | VSWRST: VSwitch domain software reset  
Set and reset by software.  
0: reset not activated (default after Backup domain reset)  
1: resets the entire VSW domain |
| 15  | RTCEN: RTC clock enable  
Set and reset by software.  
0: rtc_ck disabled (default after Backup domain reset)  
1: rtc_ck enabled |
| 14:10 | Reserved, must be kept at reset value. |
| 9:8 | RTCSEL[1:0]: RTC clock source selection  
Set by software to select the clock source for the RTC. These bits can be written only one time (except in case of failure detection on LSE). These bits must be written before LSECSSON is enabled. The VSWRST bit can be used to reset them, then it can be written one time again.  
If HSE is selected as RTC clock, this clock is lost when the system is in Stop mode or in case of a pin reset (NRST).  
00: no clock (default after Backup domain reset)  
01: LSE selected as RTC clock  
10: LSI selected as RTC clock  
11: HSE divided by RTCMPRE value selected as RTC clock |

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>LSECSSD</td>
</tr>
<tr>
<td>6</td>
<td>LSECSSON</td>
</tr>
<tr>
<td>5</td>
<td>LSEDRV[1:0]</td>
</tr>
<tr>
<td>4</td>
<td>LSEBYP</td>
</tr>
<tr>
<td>3</td>
<td>LSERDY</td>
</tr>
<tr>
<td>2</td>
<td>LSEON</td>
</tr>
<tr>
<td>1</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bit 7  **LSEEEXT**: low-speed external clock type in Bypass mode

- Set and reset by software to select the external clock type (analog or digital).
- The external clock must be enabled with the LSEON bit, to be used by the device.
- The LSEEEXT bit can be written only if the LSE oscillator is disabled.
- 0: LSE in analog mode (default after Backup domain reset)
- 1: LSE in digital mode (do not use if RTC is active).

Bit 6  **LSECSSD**: LSE clock security system failure detection

- Set by hardware to indicate when a failure has been detected by the clock security system on the external 32 kHz oscillator.
- 0: no failure detected on 32 kHz oscillator (default after Backup domain reset)
- 1: failure detected on 32 kHz oscillator

Bit 5  **LSECSSON**: LSE clock security system enable

- Set by software to enable the clock security system on 32 kHz oscillator.
- LSECSSON must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware) and after RTCSEL is selected.
- Once enabled, this bit cannot be disabled, except after a LSE failure detection (LSECSSD = 1). In that case the software must disable LSECSSON.
- 0: CSS on 32 kHz oscillator OFF (default after Backup domain reset)
- 1: CSS on 32 kHz oscillator ON

Bits 4:3  **LSEDRV[1:0]**: LSE oscillator driving capability

- Set by software to select the driving capability of the LSE oscillator.
- 00: lowest drive (default after Backup domain reset)
- 01: medium-low drive
- 10: medium-high drive
- 11: highest drive

Bit 2  **LSEBYP**: LSE oscillator bypass

- Set and reset by software to bypass oscillator in debug mode. This bit must not be written when the LSE is enabled (by LSEON) or ready (LSERDY = 1)
- 0: LSE oscillator not bypassed (default after Backup domain reset)
- 1: LSE oscillator bypassed

Bit 1  **LSERDY**: LSE oscillator ready

- Set and reset by hardware to indicate when the LSE is stable. This bit needs 6 cycles of lse_ck clock to fall down after LSEON has been set to 0.
- 0: LSE oscillator not ready (default after Backup domain reset)
- 1: LSE oscillator ready

Bit 0  **LSEON**: LSE oscillator enabled

- Set and reset by software.
- 0: LSE oscillator OFF (default after Backup domain reset)
- 1: LSE oscillator ON
8.7.25 **RCC clock control and status register (RCC_CSR)**

Address offset: 0x074  
Reset value: 0x0000 0000  
Access: 0 ≤ wait state ≤ 7, word, half-word and byte access  
Wait states are inserted in case of successive accesses to this register.

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

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

**Bit 1 LSIRDY**: LSI oscillator ready  
Set and reset by hardware to indicate when the low-speed internal RC oscillator is stable.  
This bit needs 3 cycles of lsi_ck clock to fall down after LSION has been set to 0.  
This bit can be set even when LSION is not enabled if there is a request for LSI clock by the clock security system on LSE or by the low-speed watchdog or by the RTC.  
0: LSI clock is not ready (default after reset)  
1: LSI clock is ready

**Bit 0 LSION**: LSI oscillator enable  
Set and reset by software.  
0: LSI is OFF (default after reset)  
1: LSI is ON

8.7.26 **RCC AHB3 reset register (RCC_AHB3RSTR)**

Address offset: 0x07C  
Reset value: 0x0000 0000

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

**RCC_AHB3RSTR**

<table>
<thead>
<tr>
<th>GFXMMURST</th>
<th>OCTOSPI1RST</th>
<th>OCTOSPI2RST</th>
<th>SDMMC1RST</th>
<th>OCTOSPI1RST</th>
<th>OCTOSPI2RST</th>
<th>JPDGECRST</th>
<th>DMA2CSRST</th>
<th>MCIARST</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:25  Reserved, must be kept at reset value.

Bit 24  **GFXMMURST**: GFXMMU reset  
Set and reset by software  
0: does not reset the GFXMMU (default after reset)  
1: resets the GFXMMU

Bit 23  **OTFD2RST**: OTFD2 reset  
Set and reset by software  
Take care that resetting the OTFD means loosing the decryption key loaded during secure boot.  
0: does not reset the OTFD2 (default after reset)  
1: resets the OTFD2

Bit 22  **OTFD1RST**: OTFD1 reset  
Set and reset by software  
Take care that resetting the OTFD means loosing the decryption key loaded during secure boot.  
0: does not reset the OTFD1 (default after reset)  
1: resets the OTFD1

Bit 21  **IOMNGRRST**: IOMNGR reset  
Set and reset by software  
0: does not reset the IOMNGR (default after reset)  
1: resets the IOMNGR

Bit 20  Reserved, must be kept at reset value.

Bit 19  **OCTOSPI2RST**: OCTOSPI2 and OCTOSPI2 delay block reset  
Set and reset by software  
0: does not reset the OCTOSPI2 and OCTOSPI2 delay block (default after reset)  
1: resets the OCTOSPI2 and OCTOSPI2 delay block

Bits 18:17  Reserved, must be kept at reset value.

Bit 16  **SDMMC1RST**: SDMMC1 and SDMMC1 delay blocks reset  
Set and reset by software.  
0: does not reset SDMMC1 and SDMMC1 delay blocks (default after reset)  
1: resets SDMMC1 and SDMMC1 delay blocks

Bit 15  Reserved, must be kept at reset value.

Bit 14  **OCTOSPI1RST**: OCTOSPI1 and OCTOSPI1 delay blocks reset  
Set and reset by software.  
0: does not reset OCTOSPI1 and OCTOSPI1 delay blocks (default after reset)  
1: resets OCTOSPI1 and OCTOSPI1 delay blocks

Bit 13  Reserved, must be kept at reset value.

Bit 12  **FMCRRST**: FMC block reset  
Set and reset by software.  
0: does not reset FMC block (default after reset)  
1: resets FMC block

Bits 11:6  Reserved, must be kept at reset value.

Bit 5  **JPGDECRST**: JPGDEC block reset  
Set and reset by software.  
0: does not reset JPGDEC block (default after reset)  
1: resets JPGDEC block
8.7.27 RCC AHB1 peripheral reset register (RCC_AHB1RSTR)

Address offset: 0x080
Reset value: 0x0000 0000

Bit 4  **DMA2DRST**: DMA2D block reset
Set and reset by software.
0: does not reset DMA2D block (default after reset)
1: resets DMA2D block

Bits 3:1 Reserved, must be kept at reset value.

Bit 0  **MDMARST**: MDMA block reset
Set and reset by software.
0: does not reset MDMA block (default after reset)
1: resets MDMA block

Bits 31:26 Reserved, must be kept at reset value.

Bit 25  **USB1OTGRST**: USB1OTG block reset
Set and reset by software.
0: does not reset USB1OTG block (default after reset)
1: resets USB1OTG block

Bits 24:10 Reserved, must be kept at reset value.

Bit 9  **CRCRST**: CRC block reset
Set and reset by software.
0: does not reset CRC block (default after reset)
1: resets CRC block

Bits 8:6 Reserved, must be kept at reset value.

Bit 5  **ADC12RST**: ADC1 and 2 blocks reset
Set and reset by software.
0: does not reset ADC1 and 2 blocks (default after reset)
1: resets ADC1 and 2 blocks
8.7.28 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x084

Reset value: 0x0000 0000

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

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 BDMA1RST: BDMA1 reset (DFSDM dedicated DMA)
Set and reset by software.
0: does not reset DMA block (default after reset)
1: resets DMA block

Bit 10 Reserved, must be kept at reset value.

Bit 9 SDMMC2RST: SDMMC2 and SDMMC2 delay blocks reset
Set and reset by software.
0: does not reset SDMMC2 and SDMMC2 delay blocks (default after reset)
1: resets SDMMC2 and SDMMC2 delay blocks

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 RNGRST: random number generator block reset
Set and reset by software.
0: does not reset RNG block (default after reset)
1: resets RNG block

Bit 5 HASHRST: hash block reset
Set and reset by software.
0: does not reset hash block (default after reset)
1: resets hash block
Bit 4  **CRYPTRST**: cryptography block reset  
Set and reset by software.  
0: does not reset cryptography block (default after reset)  
1: resets cryptography block  

Bit 3  Reserved, must be kept at reset value.  

Bit 2  **HSEMRST**: HSEM block reset  
Set and reset by software.  
0: does not reset HSEM block (default after reset)  
1: resets HSEM block  

Bit 1  Reserved, must be kept at reset value.  

Bit 0  **DCMI_PSSIRST**: digital camera interface block reset (DCMI or PSSI depending which IP is active)  
Set and reset by software.  
0: does not reset the DCMI/PSSI block (default after reset)  
1: resets the DCMI/PSSI block

### 8.7.29  RCC AHB4 peripheral reset register (RCC_AHB4RSTR)

Address offset: 0x088  
Reset value: 0x0000 0000

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

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

Bit 21  **BDMA2RST**: SmartRun domain DMA and DMAMUX blocks reset  
Set and reset by software.  
0: does not reset the DMA and DMAMUX blocks (default after reset)  
1: resets the DMA and DMAMUX blocks  

Bits 20:11  Reserved, must be kept at reset value.  

Bit 10  **GPIOKRST**: GPIOK block reset  
Set and reset by software.  
0: does not reset the GPIOK block (default after reset)  
1: resets the GPIOK block

Bit 9  **GPIOJRST**: GPIOJ block reset  
Set and reset by software.  
0: does not reset the GPIOJ block (default after reset)  
1: resets the GPIOJ block
Bit 8 **GPIOIRST**: GPIOI block reset
Set and reset by software.
0: does not reset the GPIOI block (default after reset)
1: resets the GPIOI block

Bit 7 **GPIOHRST**: GPIOH block reset
Set and reset by software.
0: does not reset the GPIOH block (default after reset)
1: resets the GPIOH block

Bit 6 **GPIOGRST**: GPIOG block reset
Set and reset by software.
0: does not reset the GPIOG block (default after reset)
1: resets the GPIOG block

Bit 5 **GPIOFRST**: GPIOF block reset
Set and reset by software.
0: does not reset the GPIOF block (default after reset)
1: resets the GPIOF block

Bit 4 **GPIOERST**: GPIOE block reset
Set and reset by software.
0: does not reset the GPIOE block (default after reset)
1: resets the GPIOE block

Bit 3 **GPIODRST**: GPIOD block reset
Set and reset by software.
0: does not reset the GPIOD block (default after reset)
1: resets the GPIOD block

Bit 2 **GPIOCRST**: GPIOC block reset
Set and reset by software.
0: does not reset the GPIOC block (default after reset)
1: resets the GPIOC block

Bit 1 **GPIOBRST**: GPIOB block reset
Set and reset by software.
0: does not reset the GPIOB block (default after reset)
1: resets the GPIOB block

Bit 0 **GPIOARST**: GPIOA block reset
Set and reset by software.
0: does not reset the GPIOA block (default after reset)
1: resets the GPIOA block
### 8.7.30 RCC APB3 peripheral reset register (RCC_APB3RSTR)

Address offset: 0x08C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>UART8RST</td>
<td>UART8 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>30</td>
<td>UART7RST</td>
<td>UART7 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>29</td>
<td>DAC1RST</td>
<td>DAC1 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>28</td>
<td>HDMI/ICE_RST</td>
<td>HDMI/ICE block reset</td>
<td>RW</td>
</tr>
<tr>
<td>27</td>
<td>I2C3RST</td>
<td>I2C3 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>26</td>
<td>I2C2RST</td>
<td>I2C2 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>25</td>
<td>I2C1RST</td>
<td>I2C1 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>24</td>
<td>UART5RST</td>
<td>UART5 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>23</td>
<td>UART4RST</td>
<td>UART4 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>22</td>
<td>UART3RST</td>
<td>UART3 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>21</td>
<td>UART2RST</td>
<td>UART2 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>20</td>
<td>SPI6RST</td>
<td>SPI6 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>19</td>
<td>SPI5RST</td>
<td>SPI5 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>18</td>
<td>SPI4RST</td>
<td>SPI4 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>17</td>
<td>SPI3RST</td>
<td>SPI3 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>16</td>
<td>SPI2RST</td>
<td>SPI2 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>15</td>
<td>LPTIM1RST</td>
<td>LPTIM1 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>14</td>
<td>TIM14RST</td>
<td>TIM14 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>13</td>
<td>TIM13RST</td>
<td>TIM13 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>12</td>
<td>TIM12RST</td>
<td>TIM12 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>11</td>
<td>TIM11RST</td>
<td>TIM11 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>10</td>
<td>TIM10RST</td>
<td>TIM10 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>9</td>
<td>TIM9RST</td>
<td>TIM9 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>8</td>
<td>TIM8RST</td>
<td>TIM8 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>7</td>
<td>TIM7RST</td>
<td>TIM7 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>6</td>
<td>TIM6RST</td>
<td>TIM6 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>5</td>
<td>TIM5RST</td>
<td>TIM5 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>4</td>
<td>TIM4RST</td>
<td>TIM4 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>3</td>
<td>TIM3RST</td>
<td>TIM3 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>2</td>
<td>TIM2RST</td>
<td>TIM2 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>1</td>
<td>TIM1RST</td>
<td>TIM1 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>0</td>
<td>LTDCRST</td>
<td>LTDC block reset</td>
<td>RW</td>
</tr>
</tbody>
</table>

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

Bit 3 **LTDCRST**: LTDC block reset
- Set and reset by software.
- 0: does not reset the LTDC block (default after reset)
- 1: resets the LTDC block

Bits 2:0 Reserved, must be kept at reset value.

### 8.7.31 RCC APB1 peripheral reset register (RCC_APB1LRSTR)

Address offset: 0x090  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>SPI2RST</td>
<td>SPI2 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>10</td>
<td>SPI1RST</td>
<td>SPI1 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>9</td>
<td>SPI0RST</td>
<td>SPI0 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>8</td>
<td>LMPCRST</td>
<td>LMPC block reset</td>
<td>RW</td>
</tr>
<tr>
<td>7</td>
<td>TIM12RST</td>
<td>TIM12 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>6</td>
<td>TIM11RST</td>
<td>TIM11 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>5</td>
<td>TIM10RST</td>
<td>TIM10 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>4</td>
<td>TIM9RST</td>
<td>TIM9 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>3</td>
<td>TIM8RST</td>
<td>TIM8 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>2</td>
<td>TIM7RST</td>
<td>TIM7 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>1</td>
<td>TIM6RST</td>
<td>TIM6 block reset</td>
<td>RW</td>
</tr>
<tr>
<td>0</td>
<td>TIM5RST</td>
<td>TIM5 block reset</td>
<td>RW</td>
</tr>
</tbody>
</table>

Bits 11:4 Reserved, must be kept at reset value.

Bit 3 **SPI2RST**: SPI2 block reset
- Set and reset by software.
- 0: does not reset the SPI2 block (default after reset)
- 1: resets the SPI2 block

Bits 2:0 Reserved, must be kept at reset value.
Bit 31  **UART8RST**: UART8 block reset
Set and reset by software.
0: does not reset the UART8 block (default after reset)
1: resets the UART8 block

Bit 30  **UART7RST**: UART7 block reset
Set and reset by software.
0: does not reset the UART7 block (default after reset)
1: resets the UART7 block

Bit 29  **DAC1RST**: DAC1 (containing two converters) reset
Set and reset by software.
0: does not reset the DAC1 (default after reset)
1: resets the DAC1

Bit 28  Reserved, must be kept at reset value.

Bit 27  **HDMICECRST**: HDMI-CEC block reset
Set and reset by software.
0: does not reset the HDMI-CEC block (default after reset)
1: resets the HDMI-CEC block

Bits 26:24  Reserved, must be kept at reset value.

Bit 23  **I2C3RST**: I2C3 block reset
Set and reset by software.
0: does not reset the I2C3 block (default after reset)
1: resets the I2C3 block

Bit 22  **I2C2RST**: I2C2 block reset
Set and reset by software.
0: does not reset the I2C2 block (default after reset)
1: resets the I2C2 block

Bit 21  **I2C1RST**: I2C1 block reset
Set and reset by software.
0: does not reset the I2C1 block (default after reset)
1: resets the I2C1 block

Bit 20  **UART5RST**: UART5 block reset
Set and reset by software.
0: does not reset the UART5 block (default after reset)
1: resets the UART5 block

Bit 19  **UART4RST**: UART4 block reset
Set and reset by software.
0: does not reset the UART4 block (default after reset)
1: resets the UART4 block

Bit 18  **USART3RST**: USART3 block reset
Set and reset by software.
0: does not reset the USART3 block (default after reset)
1: resets the USART3 block

Bit 17  **USART2RST**: USART2 block reset
Set and reset by software.
0: does not reset the USART2 block (default after reset)
1: resets the USART2 block
Bit 16 **SPDIFRXRST**: SPDIFRX block reset
Set and reset by software.
0: does not reset the SPDIFRX block (default after reset)
1: resets the SPDIFRX block

Bit 15 **SPI3RST**: SPI3 block reset
Set and reset by software.
0: does not reset the SPI3 block (default after reset)
1: resets the SPI3 block

Bit 14 **SPI2RST**: SPI2 block reset
Set and reset by software.
0: does not reset the SPI2 block (default after reset)
1: resets the SPI2 block

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1RST**: LPTIM1 block reset
Set and reset by software.
0: does not reset the LPTIM1 block (default after reset)
1: resets the LPTIM1 block

Bit 8 **TIM14RST**: TIM14 block reset
Set and reset by software.
0: does not reset the TIM14 block (default after reset)
1: resets the TIM14 block

Bit 7 **TIM13RST**: TIM13 block reset
Set and reset by software.
0: does not reset the TIM13 block (default after reset)
1: resets the TIM13 block

Bit 6 **TIM12RST**: TIM12 block reset
Set and reset by software.
0: does not reset the TIM12 block (default after reset)
1: resets the TIM12 block

Bit 5 **TIM7RST**: TIM7 block reset
Set and reset by software.
0: does not reset the TIM7 block (default after reset)
1: resets the TIM7 block

Bit 4 **TIM6RST**: TIM6 block reset
Set and reset by software.
0: does not reset the TIM6 block (default after reset)
1: resets the TIM6 block

Bit 3 **TIM5RST**: TIM5 block reset
Set and reset by software.
0: does not reset the TIM5 block (default after reset)
1: resets the TIM5 block
### 8.7.32 RCC APB1 peripheral reset register (RCC_APB1HRSTR)

**Address offset:** 0x094  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Default Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res.</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>Res.</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>Res.</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>Res.</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>Res.</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>Res.</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>Res.</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>Res.</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>Res.</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>Res.</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>FDCANRST</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>MDIOSRST</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>OPAMPRST</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>SWPRST</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>CRSRST</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>Res.</td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:9 Reserved, must be kept at reset value.

**Bit 8** FDCANRST: FDCAN block reset  
Set and reset by software.  
0: does not reset the FDCAN block (default after reset)  
1: resets the FDCAN block

Bits 7:6 Reserved, must be kept at reset value.

**Bit 5** MDIOSRST: MDIOS block reset  
Set and reset by software.  
0: does not reset the MDIOS block (default after reset)  
1: resets the MDIOS block

**Bit 4** OPAMPRST: OPAMP block reset  
Set and reset by software.  
0: does not reset the OPAMP block (default after reset)  
1: resets the OPAMP block

**Bit 3** Reserved, must be kept at reset value.
8.7.33 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x098

Reset value: 0x0000 0000

| Bit 31 | Reserved, must be kept at reset value. |
| Bit 30 | DFSDM1RST: DFSDM1 block reset |
| Bit 29 | Reserved, must be kept at reset value. |
| Bit 28 | SAI2RST: SAI2 block reset |
| Bit 27 | Reserved, must be kept at reset value. |
| Bit 26 | SAI1RST: SAI1 block reset |
| Bit 25 | Reserved, must be kept at reset value. |
| Bit 24 | SPI5RST: SPI5 block reset |
| Bit 23 | Reserved, must be kept at reset value. |
| Bit 22 | TIM17RST |
| Bit 21 | Reserved, must be kept at reset value. |
| Bit 20 | TIM16RST |
| Bit 19 | Reserved, must be kept at reset value. |
| Bit 18 | TIM15RST |
| Bit 17 | Reserved, must be kept at reset value. |
| Bit 16 | Reserved, must be kept at reset value. |
| Bit 15 | Reserved, must be kept at reset value. |
| Bit 14 | Reserved, must be kept at reset value. |
| Bit 13 | Reserved, must be kept at reset value. |
| Bit 12 | Reserved, must be kept at reset value. |
| Bit 11 | Reserved, must be kept at reset value. |
| Bit 10 | Reserved, must be kept at reset value. |
| Bit 9 | Reserved, must be kept at reset value. |
| Bit 8 | Reserved, must be kept at reset value. |
| Bit 7 | Reserved, must be kept at reset value. |
| Bit 6 | Reserved, must be kept at reset value. |
| Bit 5 | Reserved, must be kept at reset value. |
| Bit 4 | Reserved, must be kept at reset value. |
| Bit 3 | Reserved, must be kept at reset value. |
| Bit 2 | Reserved, must be kept at reset value. |
| Bit 1 | Reserved, must be kept at reset value. |
| Bit 0 | Reserved, must be kept at reset value. |

Bit 2 SWPRST: SWP block reset
Set and reset by software.
0: does not reset the SWP block (default after reset)
1: resets the SWP block

Bit 1 CRSRST: clock recovery system reset
Set and reset by software.
0: does not reset CRS (default after reset)
1: resets CRS

Bit 0 Reserved, must be kept at reset value.
Bit 19  Reserved, must be kept at reset value.

Bit 18  **TIM17RST**: TIM17 block reset
Set and reset by software.
0: does not reset the TIM17 block (default after reset)
1: resets the TIM17 block

Bit 17  **TIM16RST**: TIM16 block reset
Set and reset by software.
0: does not reset the TIM16 block (default after reset)
1: resets the TIM16 block

Bit 16  **TIM15RST**: TIM15 block reset
Set and reset by software.
0: does not reset the TIM15 block (default after reset)
1: resets the TIM15 block

Bits 15:14  Reserved, must be kept at reset value.

Bit 13  **SPI4RST**: SPI4 block reset
Set and reset by software.
0: does not reset the SPI4 block (default after reset)
1: resets the SPI4 block

Bit 12  **SPI1RST**: SPI1 block reset
Set and reset by software.
0: does not reset the SPI1 block (default after reset)
1: resets the SPI1 block

Bits 11:8  Reserved, must be kept at reset value.

Bit 7  **USART10RST**: USART10 block reset
Set and reset by software.
0: does not reset the USART10 block (default after reset)
1: resets the USART10 block

Bit 6  **UART9RST**: UART9 block reset
Set and reset by software.
0: does not reset the UART9 block (default after reset)
1: resets the UART9 block

Bit 5  **USART6RST**: USART6 block reset
Set and reset by software.
0: does not reset the USART6 block (default after reset)
1: resets the USART6 block

Bit 4  **USART1RST**: USART1 block reset
Set and reset by software.
0: does not reset the USART1 block (default after reset)
1: resets the USART1 block
**RCC APB4 peripheral reset register (RCC_APB4RSTR)**

Address offset: 0x09C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:28</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 27</td>
<td><strong>DFSDM2RST</strong>: DFSDM2 block reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the DFSDM2 block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the DFSDM2 block</td>
</tr>
<tr>
<td>Bit 26</td>
<td><strong>DTSRST</strong>: Digital temperature sensor block reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the DTS block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the DTS block</td>
</tr>
<tr>
<td>Bit 25:16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 15</td>
<td><strong>VREFRST</strong>: VREF block reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the VREF block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the VREF block</td>
</tr>
<tr>
<td>Bit 14</td>
<td><strong>COMP12RST</strong>: COMP1 and 2 blocks reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the COMP1 and 2 blocks (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the COMP1 and 2 blocks</td>
</tr>
</tbody>
</table>

**8.7.34 RCC APB4 peripheral reset register (RCC_APB4RSTR)**

Address offset: 0x09C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:28</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 27</td>
<td><strong>DFSDM2RST</strong>: DFSDM2 block reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the DFSDM2 block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the DFSDM2 block</td>
</tr>
<tr>
<td>Bit 26</td>
<td><strong>DTSRST</strong>: Digital temperature sensor block reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the DTS block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the DTS block</td>
</tr>
<tr>
<td>Bits 25:16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 15</td>
<td><strong>VREFRST</strong>: VREF block reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the VREF block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the VREF block</td>
</tr>
<tr>
<td>Bit 14</td>
<td><strong>COMP12RST</strong>: COMP1 and 2 blocks reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the COMP1 and 2 blocks (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the COMP1 and 2 blocks</td>
</tr>
</tbody>
</table>
Bit 13 **DAC2RST**: DAC2 (containing one converter) reset

Set and reset by software.

0: does not reset the DAC2 (default after reset)
1: resets the DAC2

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **LPTIM3RST**: LPTIM3 block reset

Set and reset by software.

0: does not reset the LPTIM3 block (default after reset)
1: resets the LPTIM3 block

Bit 9 **LPTIM2RST**: LPTIM2 block reset

Set and reset by software.

0: does not reset the LPTIM2 block (default after reset)
1: resets the LPTIM2 block

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4RST**: I2C4 block reset

Set and reset by software.

0: does not reset the I2C4 block (default after reset)
1: resets the I2C4 block

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6RST**: SPI6 block reset

Set and reset by software.

0: does not reset the SPI6 block (default after reset)
1: resets the SPI6 block

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1RST**: LPUART1 block reset

Set and reset by software.

0: does not reset the LPUART1 block (default after reset)
1: resets the LPUART1 block

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGRST**: SYSCFG block reset

Set and reset by software.

0: does not reset the SYSCFG block (default after reset)
1: resets the SYSCFG block

Bit 0 Reserved, must be kept at reset value.
8.7.35 RCC SmartRun domain autonomous mode register (RCC_SRDAMR)

The autonomous mode allows the peripheral clocks to be provided to peripherals located in the SmartRun domain, even if the CPU is in CStop mode.

When a peripheral is enabled and has its autonomous bit activated, it receives its peripheral clocks according to the SmartRun domain state, if the CPU is in CStop mode.

Address offset: 0x0A8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>Bit 31:30 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bit 29 SRDSRAMAMEN: SmartRun domain SRAM autonomous mode enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: SRDSRAM clock disabled when the CPU is in CStop (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: SRDSRAM bus clock enabled when the SmartRun domain is in Run.</td>
</tr>
<tr>
<td></td>
<td>Refer to Section 8.5.11: Peripheral clock gating control for additional information.</td>
</tr>
<tr>
<td></td>
<td>Bit 28 BKPRAMAMEN: Backup RAM autonomous mode enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: Backup RAM clock disabled when the CPU is in CStop (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: Backup RAM clock enabling is controlled by the SmartRun domain state.</td>
</tr>
<tr>
<td></td>
<td>Refer to Section 8.5.11: Peripheral clock gating control for additional information.</td>
</tr>
<tr>
<td></td>
<td>Bit 27 DFSDM2AMEN: DFSDM2 autonomous mode enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: DFSDM2 clock disabled when the CPU is in CStop (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: DFSDM2 peripheral clocks enabled when the SmartRun domain is in Run mode. Kernel clock enabled when the SmartRun domain is in Stop mode.</td>
</tr>
<tr>
<td></td>
<td>Refer to Section 8.5.11: Peripheral clock gating control for additional information.</td>
</tr>
<tr>
<td></td>
<td>Bit 26 DTSAMEN: Digital temperature sensor autonomous mode enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: DTS clocks disabled when the CPU is in CStop (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: DTS clocks enabled when the SmartRun domain is in Run.</td>
</tr>
<tr>
<td></td>
<td>Refer to Section 8.5.11: Peripheral clock gating control for additional information.</td>
</tr>
<tr>
<td></td>
<td>Bits 25:17 Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bit 16 **RTC\_AMEN**: RTC autonomous mode enable

- Set and reset by software.
- 0: RTC bus clocks disabled when the CPU is in CStop (default after reset)
- 1: RTC bus clocks enabled when the SmartRun domain is in Run.

Refer to Section 8.5.11: Peripheral clock gating control for additional information.

Bit 15 **V\_REF\_AMEN**: V\_REF autonomous mode enable

- Set and reset by software.
- 0: V\_REF clocks disabled when the CPU is in CStop (default after reset)
- 1: V\_REF clocks enabled when the SmartRun domain is in Run or Stop mode.

Refer to Section 8.5.11: Peripheral clock gating control for additional information.

Bit 14 **COMP\_1\_2\_AMEN**: COMP1 and 2 autonomous mode enable

- Set and reset by software.
- 0: COMP1 and 2 peripheral clocks disabled when the CPU is in CStop (default after reset)
- 1: COMP1 and 2 peripheral clocks enabled when the SmartRun domain is in Run.

Refer to Section 8.5.11: Peripheral clock gating control for additional information.

Bit 13 **DAC\_2\_AMEN**: DAC2 (containing one converter) autonomous mode enable

- Set and reset by software.
- 0: DAC2 peripheral clocks disabled when the CPU is in CStop (default after reset)
- 1: DAC2 peripheral clocks enabled when the SmartRun domain is in Run.

Refer to Section 8.5.11: Peripheral clock gating control for additional information.

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **LPTIM\_3\_AMEN**: LPTIM3 autonomous mode enable

- Set and reset by software.
- 0: LPTIM3 peripheral clocks disabled when the CPU is in CStop (default after reset)
- 1: LPTIM3 peripheral clocks enabled when the SmartRun domain is in Run mode. Kernel clock is enabled when the SmartRun domain is in Stop mode.

Refer to Section 8.5.11: Peripheral clock gating control for additional information.

Bit 9 **LPTIM\_2\_AMEN**: LPTIM2 autonomous mode enable

- Set and reset by software.
- 0: LPTIM2 peripheral clocks are disabled when the CPU is in CStop (default after reset)
- 1: LPTIM2 peripheral clocks enabled when the SmartRun domain is in Run mode. Kernel clock is enabled when the SmartRun domain is in Stop mode.

Refer to Section 8.5.11: Peripheral clock gating control for additional information.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C\_4\_AMEN**: I2C4 autonomous mode enable

- Set and reset by software.
- 0: I2C4 peripheral clocks disabled when the CPU is in CStop (default after reset)
- 1: I2C4 peripheral clocks enabled when the SmartRun domain is in Run mode. Kernel clock is enabled when the SmartRun domain is in Stop mode.

Refer to Section 8.5.11: Peripheral clock gating control for additional information.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI\_6\_AMEN**: SPI6 autonomous mode enable

- Set and reset by software.
- 0: SPI6 peripheral clocks disabled when the CPU is in CStop (default after reset)
- 1: SPI6 peripheral clocks enabled when the SmartRun domain is in Run mode. Kernel clock is enabled when the SmartRun domain is in Stop mode.

Refer to Section 8.5.11: Peripheral clock gating control for additional information.
Bit 4  Reserved, must be kept at reset value.

Bit 3  **LPUART1AMEN:** LPUART1 autonomous mode enable
       Set and reset by software.
       0: LPUART1 peripheral clocks disabled when the CPU is in CStop (default after reset)
       1: LPUART1 peripheral clocks enabled when the SmartRun domain is in Run mode. Kernel clock is enabled when the SmartRun domain is in Stop mode.
       Refer to [Section 8.5.11: Peripheral clock gating control](#) for additional information.

Bit 2  Reserved, must be kept at reset value.

Bit 1  **GPIOAMEN:** GPIO autonomous mode enable
       Set and reset by software.
       0: GPIO peripheral clocks disabled when the CPU is in CStop (default after reset)
       1: GPIO peripheral clocks enabled when the SmartRun domain is in Run.
       Refer to [Section 8.5.11: Peripheral clock gating control](#) for additional information.

Bit 0  **BDMA2AMEN:** SmartRun domain DMA and DMAMUX autonomous mode enable
       Set and reset by software.
       0: BDMA2 and DMAMUX2 peripheral clocks disabled when the CPU is in CStop (default after reset)
       1: BDMA2 and DMAMUX2 peripheral clocks enabled when the SmartRun domain is in Run.
       Refer to [Section 8.5.11: Peripheral clock gating control](#) for additional information.
### 8.7.36 RCC AXI clocks gating enable register (RCC_CKGAENR)

Address offset: 0x0B0  
Reset value: 0x0000 0000

The dynamic power consumption can be optimized by enabling the functional clock gating.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>JTAGCKG: JTAG automatic clock gating</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and reset by software.</td>
</tr>
<tr>
<td>0</td>
<td>The clock gating is disabled. The clock is always enabled.</td>
</tr>
<tr>
<td>1</td>
<td>The clock gating is enabled. The clock is disabled except if a JTAG connection has been detected</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>EXTICKG: EXTI clock gating</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and reset by software.</td>
</tr>
<tr>
<td>0</td>
<td>The clock gating is disabled. The clock is always enabled.</td>
</tr>
<tr>
<td>1</td>
<td>The clock gating is enabled. The clock is enabled after an event detection and stopped again when the event flag is cleared.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>ECCRAMCKG: RAM error code correction (ECC) clock gating</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and reset by software.</td>
</tr>
<tr>
<td>0</td>
<td>The clock gating is disabled. The clock is always enabled.</td>
</tr>
<tr>
<td>1</td>
<td>The clock gating is enabled. The ECC clock is enabled only during a RAM access.</td>
</tr>
</tbody>
</table>

| Bits 28:18 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bit 17</th>
<th>GFXMMUSCKG: AXI matrix slave GFXMMU clock gating</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and reset by software.</td>
</tr>
<tr>
<td>0</td>
<td>The clock gating is disabled. The clock is always enabled.</td>
</tr>
<tr>
<td>1</td>
<td>The clock gating is enabled. The AXI matrix slave GFXMMU clock is enabled on bus transaction request.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 16</th>
<th>AXIRAM3CKG: AXI matrix slave SRAM3 clock gating</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and reset by software.</td>
</tr>
<tr>
<td>0</td>
<td>The clock gating is disabled. The clock is always enabled.</td>
</tr>
<tr>
<td>1</td>
<td>The clock gating is enabled. The AXI matrix slave SRAM3 clock is enabled on bus transaction request.</td>
</tr>
</tbody>
</table>
Bit 15  **AXIRAM2CKG**: AXI matrix slave SRAM2 clock gating
        This bit is set and reset by software.
        0: The clock gating is disabled. The clock is always enabled.
        1: The clock gating is enabled. The AXI matrix slave SRAM2 clock is enabled on bus transaction request.

Bit 14  **AXIRAM1CKG**: AXI slave SRAM1 clock gating
        This bit is set and reset by software.
        0: The clock gating is disabled. The clock is always enabled.
        1: The clock gating is enabled. The AXI matrix slave SRAM1 clock is enabled on bus transaction request.

Bit 13  **OCTOSPI1CKG**: AXI slave OCTOSPI1 clock gating
        This bit is set and reset by software.
        0: The clock gating is disabled. The clock is always enabled.
        1: The clock gating is enabled. The AXI matrix slave OCTOSPI1 clock is enabled on bus transaction request.

Bit 12  **FMCCKG**: AXI slave FMC clock gating
        This bit is set and reset by software.
        0: The clock gating is disabled. The clock is always enabled.
        1: The clock gating is enabled. The AXI matrix slave FMC clock is enabled on bus transaction request.

Bit 11  **OCTOSPI2CKG**: AXI slave OCTOSPI2 clock gating
        This bit is set and reset by software.
        0: The clock gating is disabled. The clock is always enabled.
        1: The clock gating is enabled. The AXI matrix slave OCTOSPI2 clock is enabled on bus transaction request.

Bit 10  **FLIFTCKG**: AXI slave Flash interface (FLIFT) clock gating
        This bit is set and reset by software.
        0: The clock gating is disabled. The clock is always enabled.
        1: The clock gating is enabled. The AXI matrix slave FLIFT clock is enabled on bus transaction request.

Bit 9   **AHB34CKG**: AXI slave AHB34 clock gating
        This bit is set and reset by software.
        0: The clock gating is disabled. The clock is always enabled.
        1: The clock gating is enabled. The AXI matrix slave AHB34 clock is enabled on bus transaction request.

Bit 8   **AHB12CKG**: AXI slave AHB12 clock gating
        This bit is set and reset by software.
        0: The clock gating is disabled. The clock is always enabled.
        1: The clock gating is enabled. The AXI matrix slave AHB12 clock is enabled on bus transaction request.

Bit 7   **GFXMMUMCKG**: AXI master GFXMMU clock gating
        This bit is set and reset by software.
        0: The clock gating is disabled. The clock is always enabled.
        1: The clock gating is enabled. The AXI master GFXMMU clock is enabled on bus transaction request.
Bit 6  **LTDCCKG**: AXI master LTDC clock gating
This bit is set and reset by software.
0: The clock gating is disabled. The clock is always enabled.
1: The clock gating is enabled. The AXI matrix master LTDC clock is enabled on bus transaction request.

Bit 5  **DMA2DCKG**: AXI master DMA2D clock gating
This bit is set and reset by software.
0: The clock gating is disabled. The clock is always enabled.
1: The clock gating is enabled. The AXI matrix master DMA2D clock is enabled on bus transaction request.

Bit 4  **MDMACKG**: AXI master MDMA clock gating
This bit is set and reset by software.
0: The clock gating is disabled. The clock is always enabled.
1: The clock gating is enabled. The AXI matrix master MDMA clock is enabled on bus transaction request.

Bit 3  **SDMMCCKG**: AXI master SDMMC clock gating
This bit is set and reset by software.
0: The clock gating is disabled. The clock is always enabled.
1: The clock gating is enabled. The AXI matrix master SDMMC clock is enabled on bus transaction request.

Bit 2  **CPUCKG**: AXI master CPU clock gating
This bit is set and reset by software.
0: The clock gating is disabled. The clock is always enabled.
1: The clock gating is enabled. The AXI matrix master CPU clock is enabled on bus transaction request.

Bit 1  **AHBCKG**: AXI master AHB clock gating
This bit is set and reset by software.
0: The clock gating is disabled. The clock is always enabled.
1: The clock gating is enabled. The AXI matrix master AHB clock is enabled on bus transaction request.

Bit 0  **AXICKG**: AXI interconnect matrix clock gating
This bit is set and reset by software.
0: The clock gating is disabled. The clock is always enabled.
1: The clock gating is enabled. The AXI interconnect matrix clock is enabled on bus transaction request.
8.7.37 RCC reset status register (RCC_RSR)

Address offset: 0x130
Reset value: 0x00E8 0000
Reset by power-on reset only.

Access: 0 ≤ wait state ≤ 7, word, half-word and byte access. Wait states are inserted in case of successive accesses to this register.

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LPWRRSTF**: reset due to illegal CD DStop or CD DStop2 or CPU CStop flag \(^{(1)}\)
- Reset by software by writing the RMVF bit.
- Set by hardware when the CPU domain goes erroneously in DStop or DStop2, or when the CPU goes erroneously in CStop.
- 0: no illegal reset occurred (default after power-on reset)
- 1: illegal CD DStop or CD DStop2 or CPU CStop reset occurred

Bit 29 Reserved, must be kept at reset value.

Bit 28 **WWGGRSTF**: window watchdog reset flag \(^{(1)}\)
- Reset by software by writing the RMVF bit.
- Set by hardware when a window watchdog reset occurs.
- 0: no window watchdog reset occurred from WWDG (default after power-on reset)
- 1: window watchdog reset occurred from WWDG

Bit 27 Reserved, must be kept at reset value.

Bit 26 **MGDGSTF**: independent watchdog reset flag \(^{(1)}\)
- Reset by software by writing the RMVF bit.
- Set by hardware when an independent watchdog reset occurs.
- 0: no independent watchdog reset occurred (default after power-on reset)
- 1: independent watchdog reset occurred

Bit 25 Reserved, must be kept at reset value.

Bit 24 **SFTRSTF**: system reset from CPU reset flag \(^{(1)}\)
- Reset by software by writing the RMVF bit.
- Set by hardware when the system reset is due to CPU. The CPU can generate a system reset by writing SYSRESETREQ bit of AIRCR register of the core M7.
- 0: no CPU software reset occurred (default after power-on reset)
- 1: a system reset has been generated by the CPU
Bit 23 **PORRSTF**: POR/PDR reset flag (1)
   - Reset by software by writing the RMVF bit.
   - Set by hardware when a POR/PDR reset occurs.
   - 0: no POR/PDR reset occurred
   - 1: POR/PDR reset occurred (default after power-on reset)

Bit 22 **PINRSTF**: pin reset flag (NRST) (1)
   - Reset by software by writing the RMVF bit.
   - Set by hardware when a reset from pin occurs.
   - 0: no reset from pin occurred
   - 1: reset from pin occurred (default after power-on reset)

Bit 21 **BORRSTF**: BOR reset flag (1)
   - Reset by software by writing the RMVF bit.
   - Set by hardware when a BOR reset occurs (pwr_bor_rst).
   - 0: no BOR reset occurred
   - 1: BOR reset occurred (default after power-on reset)

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CDRSTF**: CPU domain power-switch reset flag (1)
   - Reset by software by writing the RMVF bit.
   - Set by hardware when the CPU domain exits from DStop or after of power-on reset. Set also when the CPU domain exists DStop2 but only when a pad reset has occurred during DStop2 (PINRST bit also set by hardware)
   - 0: no CPU domain power-switch reset occurred
   - 1: CPU domain power-switch (ePOD2) reset occurred (default after power-on reset)

Bits 18:17 Reserved, must be kept at reset value.

Bit 16 **RMVF**: remove reset flag
   - Set and reset by software to reset the value of the reset flags.
   - 0: reset of the reset flags not activated (default after power-on reset)
   - 1: resets the value of the reset flags

Bits 15:0 Reserved, must be kept at reset value.

1. Refer to Table 54: Reset source identification (RCC_RSR) for details on flag behavior.
### 8.7.38 RCC AHB3 clock register (RCC_AHB3ENR)

Address offset: 0x134  
Reset value: 0x0000 0000

| Bit 31:25 Reserved, must be kept at reset value. |  |
| Bit 24 **GFXMMUEN**: GFXMMU clock enable  
Set and reset by software.  
0: GFXMMU clock disabled (default after reset)  
1: GFXMMU clock enabled |  |
| Bit 23 **OTFD2EN**: OTFD2 clock enable  
Set and reset by software.  
0: OTFD2 clock disabled (default after reset)  
1: OTFD2 clock enabled |  |
| Bit 22 **OTFD1EN**: OTFD1 clock enable  
Set and reset by software.  
0: OTFD1 clock disabled (default after reset)  
1: OTFD1 clock enabled |  |
| Bit 21 **IOMNGREN**: IOMNGR clock enable  
Set and reset by software.  
0: IOMNGR clock disabled (default after reset)  
1: IOMNGR clock enabled |  |
| Bit 20 Reserved, must be kept at reset value. |  |
| Bit 19 **OCTOSPI2EN**: OCTOSPI2 clock enable  
Set and reset by software.  
0: OCTOSPI2 and OCTOSPI2 delay clock disabled (default after reset)  
1: OCTOSPI2 OCTOSPI2 delay clock enabled |  |
| Bits 18:17 Reserved, must be kept at reset value. |  |
| Bit 16 **SDMMC1EN**: SDMMC1 and SDMMC1 delay clock enable  
Set and reset by software.  
0: SDMMC1 and SDMMC1 delay clock disabled (default after reset)  
1: SDMMC1 and SDMMC1 delay clock enabled |  |
| Bit 15 Reserved, must be kept at reset value. |  |
Bit 14 OCTOSPI1EN: OCTOSPI1 and OCTOSPI1 delay clock enable
   Set and reset by software.
   0: OCTOSPI1 and OCTOSPI1 delay clock disabled (default after reset)
   1: OCTOSPI1 and OCTOSPI1 delay clock enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 FMCEN: FMC peripheral clocks enable
   Set and reset by software.
   0: FMC peripheral clocks disabled (default after reset)
   1: FMC peripheral clocks enabled
   The peripheral clocks of the FMC are the kernel clock selected by FMCSEL and provided to
   KCK_FMC input, and the rcc_hclk3 bus interface clock.

Bits 11:6 Reserved, must be kept at reset value.

Bit 5 JPGDECEN: JPGDEC peripheral clock enable
   Set and reset by software.
   0: JPGDEC peripheral clock disabled (default after reset)
   1: JPGDEC peripheral clock enabled

Bit 4 DMA2DEN: DMA2D peripheral clock enable
   Set and reset by software.
   0: DMA2D peripheral clock disabled (default after reset)
   1: DMA2D peripheral clock enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 MDMAEN: MDMA peripheral clock enable
   Set and reset by software.
   0: MDMA peripheral clock disabled (default after reset)
   1: MDMA peripheral clock enabled

8.7.39 RCC AHB1 clock register (RCC_AHB1ENR)

Address offset: 0x138
Reset value: 0x0000 0000

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

440/2974  RM0455 Rev 3
Bits 31:27  Reserved, must be kept at reset value.

Bit 26 **USB1ULPIEN**: USB_PHY1 clocks enable
- Set and reset by software.
- 0: USB1ULPI PHY clocks disabled (default after reset)
- 1: USB1ULPI PHY clocks enabled

Bit 25 **USB1OTGEN**: USB1OTG peripheral clocks enable
- Set and reset by software.
- 0: USB1OTG peripheral clocks disabled (default after reset)
- 1: USB1OTG peripheral clocks enabled
The peripheral clocks of the USB1OTG are the kernel clock selected by USBSEL and the `rcc_hclk1` bus interface clock.

Bits 24:10  Reserved, must be kept at reset value.

Bit 9 **CRCEN**: CRC peripheral clock enable
- Set and reset by software.
- 0: CRC peripheral clock disabled (default after reset)
- 1: CRC peripheral clock enabled

Bits 8:6  Reserved, must be kept at reset value.

Bit 5 **ADC12EN**: ADC1 and 2 peripheral clocks enable
- Set and reset by software.
- 0: ADC1 and 2 peripheral clocks disabled (default after reset)
- 1: ADC1 and 2 peripheral clocks enabled
The peripheral clocks of the ADC1 and 2 are the kernel clock selected by ADCSEL and provided to ADCx_CK input, and the `rcc_hclk1` bus interface clock.

Bits 4:2  Reserved, must be kept at reset value.

Bit 1 **DMA2EN**: DMA2 clock enable
- Set and reset by software.
- 0: DMA2 clock disabled (default after reset)
- 1: DMA2 clock enabled

Bit 0 **DMA1EN**: DMA1 clock enable
- Set and reset by software.
- 0: DMA1 clock disabled (default after reset)
- 1: DMA1 clock enabled
### 8.7.40 RCC AHB2 clock register (RCC_AHB2ENR)

Address offset: 0x13C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28-12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8-7</th>
<th>Bit 6-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>Res.</td>
<td>Res.</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

- **Bit 31** Reserved, must be kept at reset value.
- **Bit 30** **AHBSRAM2EN**: AHBSRAM2 block enable  
  Set and reset by software.  
  When set, this bit indicates that the SRAM2 is allocated by the CPU. It causes the CPU domain to take into account also the CPU operation modes, keeping the CPU domain in DRun when the CPU is in CRun.  
  0: AHBSRAM2 interface clock is disabled. (default after reset)  
  1: AHBSRAM2 interface clock is enabled.

- **Bit 29** **AHBSRAM1EN**: AHBSRAM1 block enable  
  Set and reset by software.  
  When set, this bit indicates that the SRAM1 is allocated by the CPU. It causes the CPU domain to take into account also the CPU operation modes, keeping the CPU domain in DRun when the CPU is in CRun.  
  0: AHBSRAM1 interface clock is disabled. (default after reset)  
  1: AHBSRAM1 interface clock is enabled.

- **Bits 28-12** Reserved, must be kept at reset value.
- **Bit 11** **BDMA1EN**: DMA clock enable (DFSDM dedicated DMA)  
  Set and reset by software.  
  0: DMA clock disabled (default after reset)  
  1: DMA clock enabled

- **Bit 10** Reserved, must be kept at reset value.
- **Bit 9** **SDMMC2EN**: SDMMC2 and SDMMC2 delay clock enable  
  Set and reset by software.  
  0: SDMMC2 and SDMMC2 delay clock disabled (default after reset)  
  1: SDMMC2 and SDMMC2 delay clock enabled

- **Bits 8-7** Reserved, must be kept at reset value.
**8.7.41 RCC AHB4 Clock Register (RCC_AHB4ENR)**

Address offset: 0x140

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Default Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>SRDSRAMEN</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>BKPRAMEN</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>RES</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>RES</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>RES</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>RES</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>RES</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>RES</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>BDMAEN</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>RES</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>RES</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>RES</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>RES</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>RES</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>RES</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bit 6 **RNGEN**: RNG peripheral clocks enable
- Set and reset by software.
- 0: RNG peripheral clocks disabled (default after reset)
- 1: RNG peripheral clocks enabled.

The peripheral clocks of the RNG are the kernel clock selected by RNGSEL and provided to **rng_clk** input, and the **rcc_hclk2** bus interface clock.

Bit 5 **HASHEN**: HASH peripheral clock enable
- Set and reset by software.
- 0: HASH peripheral clock disabled (default after reset)
- 1: HASH peripheral clock enabled

Bit 4 **CRYPTEN**: CRYPT peripheral clock enable
- Set and reset by software.
- 0: CRYPT peripheral clock disabled (default after reset)
- 1: CRYPT peripheral clock enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **HSEmen**: HSEM peripheral clock enable
- Set and reset by software.
- 0: HSEM peripheral clock disabled (default after reset)
- 1: HSEM peripheral clock enabled

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DCMI_PSIEN**: digital camera interface peripheral clock enable (DCMI or PSSI depending which IP is active)
- Set and reset by software.
- 0: DCMI/PSSI peripheral clock disabled (default after reset)
- 1: DCMI/PSSI peripheral clock enabled
Bits 31:30  Reserved, must be kept at reset value.

Bit 29  **SRDSRAMEN**: SmartRun domain SRAM clock enable
      Set and reset by software.
      0: SRDSRAM clock disabled (default after reset)
      1: SRDSRAM clock enabled

Bit 28  **BKPRAMEN**: Backup RAM clock enable
      Set and reset by software.
      0: Backup RAM clock disabled (default after reset)
      1: Backup RAM clock enabled

Bits 27:22  Reserved, must be kept at reset value.

Bit 21  **BDMA2EN**: SmartRun domain DMA and DMAMUX clock enable
      Set and reset by software.
      0: BDMA2 and DMAMUX2 clock disabled (default after reset)
      1: BDMA2 and DMAMUX2 clock enabled

Bits 20:11  Reserved, must be kept at reset value.

Bit 10  **GPIOKEN**: GPIOK peripheral clock enable
      Set and reset by software.
      0: GPIOK peripheral clock disabled (default after reset)
      1: GPIOK peripheral clock enabled

Bit 9  **GPIOJEN**: GPIOJ peripheral clock enable
      Set and reset by software.
      0: GPIOJ peripheral clock disabled (default after reset)
      1: GPIOJ peripheral clock enabled

Bit 8  **GPIOIEN**: GPIOI peripheral clock enable
      Set and reset by software.
      0: GPIOI peripheral clock disabled (default after reset)
      1: GPIOI peripheral clock enabled

Bit 7  **GPIOHEN**: GPIOH peripheral clock enable
      Set and reset by software.
      0: GPIOH peripheral clock disabled (default after reset)
      1: GPIOH peripheral clock enabled

Bit 6  **GPIOGEN**: GPIOG peripheral clock enable
      Set and reset by software.
      0: GPIOG peripheral clock disabled (default after reset)
      1: GPIOG peripheral clock enabled

Bit 5  **GPIOFEN**: GPIOF peripheral clock enable
      Set and reset by software.
      0: GPIOF peripheral clock disabled (default after reset)
      1: GPIOF peripheral clock enabled

Bit 4  **GPIOEEN**: GPIOE peripheral clock enable
      Set and reset by software.
      0: GPIOE peripheral clock disabled (default after reset)
      1: GPIOE peripheral clock enabled
Bit 3  **GPIODEN**: GPIOD peripheral clock enable
- Set and reset by software.
  - 0: GPIOD peripheral clock disabled (default after reset)
  - 1: GPIOD peripheral clock enabled

Bit 2  **GPIOCEN**: GPIOC peripheral clock enable
- Set and reset by software.
  - 0: GPIOC peripheral clock disabled (default after reset)
  - 1: GPIOC peripheral clock enabled

Bit 1  **GPIOBEN**: GPIOB peripheral clock enable
- Set and reset by software.
  - 0: GPIOB peripheral clock disabled (default after reset)
  - 1: GPIOB peripheral clock enabled

Bit 0  **GPIOAEN**: GPIOA peripheral clock enable
- Set and reset by software.
  - 0: GPIOA peripheral clock disabled (default after reset)
  - 1: GPIOA peripheral clock enabled

### 8.7.42 RCC APB3 clock register (RCC_APB3ENR)

Address offset: 0x144

Reset value: 0x0000 0000

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6  **WWDGEN**: WWDG clock enable
- Set by software, and reset by hardware when a system reset occurs.
  - Note that in order to work properly, before enabling the WWDG, the bit WW1RSC must be set to 1.
  - 0: WWDG peripheral clock disable (default after reset)
  - 1: WWDG peripheral clock enabled

Bits 5:4 Reserved, must be kept at reset value.

Bit 3  **LTDCEN**: LTDC peripheral clock enable
- Provides the pixel clock (**ltdc_clk**) to the LTDC block.
  - Set and reset by software.
  - 0: LTDC peripheral clock disabled (default after reset)
  - 1: LTDC peripheral clock provided to the LTDC block

Bits 2:0 Reserved, must be kept at reset value.
8.7.43 RCC APB1 clock register (RCC_APB1LENR)

Address offset: 0x148
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UART8EN: UART8 peripheral clocks enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td>0:</td>
<td>UART8 peripheral clocks disabled (default after reset)</td>
</tr>
<tr>
<td>1:</td>
<td>UART8 peripheral clocks enabled</td>
</tr>
<tr>
<td></td>
<td>The peripheral clocks of the UART8 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the rcc_pclk1 bus interface clock.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>UART7EN: UART7 peripheral clocks enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td>0:</td>
<td>UART7 peripheral clocks disabled (default after reset)</td>
</tr>
<tr>
<td>1:</td>
<td>UART7 peripheral clocks enabled</td>
</tr>
<tr>
<td></td>
<td>The peripheral clocks of the UART7 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the rcc_pclk1 bus interface clock.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>DAC1EN: DAC1 (containing two converters) peripheral clock enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td>0:</td>
<td>DAC1 peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td>1:</td>
<td>DAC1 peripheral clock enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 27</th>
<th>HDMICECEN: HDMI-CEC peripheral clock enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td>0:</td>
<td>HDMI-CEC peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td>1:</td>
<td>HDMI-CEC peripheral clock enabled</td>
</tr>
<tr>
<td></td>
<td>The peripheral clocks of the HDMI-CEC are the kernel clock selected by CECSSEL and provided to clk_cec input, and the rcc_pclk1 bus interface clock.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 26:24</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 23</th>
<th>I2C3EN: I2C3 peripheral clocks enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td>0:</td>
<td>I2C3 peripheral clocks disabled (default after reset)</td>
</tr>
<tr>
<td>1:</td>
<td>I2C3 peripheral clocks enabled</td>
</tr>
<tr>
<td></td>
<td>The peripheral clocks of the I2C3 are the kernel clock selected by I2C123SEL and provided to I2CCLK input, and the rcc_pclk1 bus interface clock.</td>
</tr>
</tbody>
</table>
Bit 22  **I2C2EN**: I2C2 peripheral clocks enable  
Set and reset by software.  
0: I2C2 peripheral clocks disabled (default after reset)  
1: I2C2 peripheral clocks enabled  
The peripheral clocks of the I2C2 are the kernel clock selected by I2C123SEL and provided to I2CCLK input, and the `rcc_pclk1` bus interface clock.

Bit 21  **I2C1EN**: I2C1 peripheral clocks enable  
Set and reset by software.  
0: I2C1 peripheral clocks disabled (default after reset)  
1: I2C1 peripheral clocks enabled  
The peripheral clocks of the I2C1 are the kernel clock selected by I2C123SEL and provided to I2CCLK input, and the `rcc_pclk1` bus interface clock.

Bit 20  **UART5EN**: UART5 peripheral clocks enable  
Set and reset by software.  
0: UART5 peripheral clocks disabled (default after reset)  
1: UART5 peripheral clocks enabled  
The peripheral clocks of the UART5 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the `rcc_pclk1` bus interface clock.

Bit 19  **UART4EN**: UART4 peripheral clocks enable  
Set and reset by software.  
0: UART4 peripheral clocks disabled (default after reset)  
1: UART4 peripheral clocks enabled  
The peripheral clocks of the UART4 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the `rcc_pclk1` bus interface clock.

Bit 18  **USART3EN**: USART3 peripheral clocks enable  
Set and reset by software.  
0: USART3 peripheral clocks disabled (default after reset)  
1: USART3 peripheral clocks enabled  
The peripheral clocks of the USART3 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the `rcc_pclk1` bus interface clock.

Bit 17  **USART2EN**: USART2 peripheral clocks enable  
Set and reset by software.  
0: USART2 peripheral clocks disabled (default after reset)  
1: USART2 peripheral clocks enabled  
The peripheral clocks of the USART2 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the `rcc_pclk1` bus interface clock.

Bit 16  **SPDIFRXEN**: SPDIFRX peripheral clocks enable  
Set and reset by software.  
0: SPDIFRX peripheral clocks disabled (default after reset)  
1: SPDIFRX peripheral clocks enabled  
The peripheral clocks of the SPDIFRX are the kernel clock selected by SPDIFRXSEL and provided to SPDIFRX_CLK input, and the `rcc_pclk1` bus interface clock.

Bit 15  **SPI3EN**: SPI3 peripheral clocks enable  
Set and reset by software.  
0: SPI3 peripheral clocks disabled (default after reset)  
1: SPI3 peripheral clocks enabled  
The peripheral clocks of the SPI3 are the kernel clock selected by I2S123SRC and provided to com_clk input, and the `rcc_pclk1` bus interface clock.
Bit 14  **SPI2EN**: SPI2 peripheral clocks enable
       Set and reset by software.
       0: SPI2 peripheral clocks disabled (default after reset)
       1: SPI2 peripheral clocks enabled
       The peripheral clocks of the SPI2 are the kernel clock selected by I2S123SRC and provided to
       **com_clk** input, and the **rcc_pclk1** bus interface clock.

Bits 13:10 Reserved, must be kept at reset value.

Bit 9  **LPTIM1EN**: LPTIM1 peripheral clocks enable
       Set and reset by software.
       0: LPTIM1 peripheral clocks disabled (default after reset)
       1: LPTIM1 peripheral clocks enabled
       The peripheral clocks of the LPTIM1 are the kernel clock selected by LPTIM1SEL and provided to
       **clk_lpt** input, and the **rcc_pclk1** bus interface clock.

Bit 8  **TIM14EN**: TIM14 peripheral clock enable
       Set and reset by software.
       0: TIM14 peripheral clock disabled (default after reset)
       1: TIM14 peripheral clock enabled

Bit 7  **TIM13EN**: TIM13 peripheral clock enable
       Set and reset by software.
       0: TIM13 peripheral clock disabled (default after reset)
       1: TIM13 peripheral clock enabled

Bit 6  **TIM12EN**: TIM12 peripheral clock enable
       Set and reset by software.
       0: TIM12 peripheral clock disabled (default after reset)
       1: TIM12 peripheral clock enabled

Bit 5  **TIM7EN**: TIM7 peripheral clock enable
       Set and reset by software.
       0: TIM7 peripheral clock disabled (default after reset)
       1: TIM7 peripheral clock enabled

Bit 4  **TIM6EN**: TIM6 peripheral clock enable
       Set and reset by software.
       0: TIM6 peripheral clock disabled (default after reset)
       1: TIM6 peripheral clock enabled

Bit 3  **TIM5EN**: TIM5 peripheral clock enable
       Set and reset by software.
       0: TIM5 peripheral clock disabled (default after reset)
       1: TIM5 peripheral clock enabled
Bit 2 **TIM4EN**: TIM4 peripheral clock enable
   Set and reset by software.
   0: TIM4 peripheral clock disable (default after reset)
   1: TIM4 peripheral clock enabled

Bit 1 **TIM3EN**: TIM3 peripheral clock enable
   Set and reset by software.
   0: TIM3 peripheral clock disabled (default after reset)
   1: TIM3 peripheral clock enabled

Bit 0 **TIM2EN**: TIM2 peripheral clock enable
   Set and reset by software.
   0: TIM2 peripheral clock disabled (default after reset)
   1: TIM2 peripheral clock enabled

### 8.7.44 RCC APB1 clock register (RCC_APB1HENR)
Address offset: 0x14C
Reset value: 0x0000 0000

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **FDCANEN**: FDCAN peripheral clocks enable
   Set and reset by software.
   0: FDCAN peripheral clocks disabled (default after reset)
   1: FDCAN peripheral clocks enabled:
   The peripheral clocks of the FDCAN are the kernel clock selected by FDCANSEL and provided to `fdcan_clk` input, and the `rcc_pclk1` bus interface clock.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSEN**: MDIOS peripheral clock enable
   Set and reset by software.
   0: MDIOS peripheral clock disabled (default after reset)
   1: MDIOS peripheral clock enabled

Bit 4 **OPAMPEN**: OPAMP peripheral clock enable
   Set and reset by software.
   0: OPAMP peripheral clock disabled (default after reset)
   1: OPAMP peripheral clock enabled

Bit 3 Reserved, must be kept at reset value.
Bit 2 **SWPEN**: SWP peripheral clocks enable
Set and reset by software.
0: SWP peripheral clocks disabled (default after reset)
1: SWP peripheral clocks enabled.
The peripheral clocks of the SWP are: the kernel clock selected by SWPSEL and provided to SWPCLK input, and the `rcc_pclk1` bus interface clock.

Bit 1 **CRSEN**: clock recovery system peripheral clock enable
Set and reset by software.
0: CRS peripheral clock disabled (default after reset)
1: CRS peripheral clock enabled

Bit 0 Reserved, must be kept at reset value.

### 8.7.45 RCC APB2 clock register (RCC_APB2ENR)

Address offset: 0x150
Reset value: 0x0000 0000

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DFSDM1EN**: DFSDM1 peripheral clocks enable
Set and reset by software.
0: DFSDM1 peripheral clocks disabled (default after reset)
1: DFSDM1 peripheral clocks enabled
DFSDM1 peripheral clocks are the kernel clocks selected by SAI1SEL and DFSDM1SEL and provided to `Aclk` and `clk` inputs respectively.

Bits 29:24 Reserved, must be kept at reset value.

Bit 23 **SAI2EN**: SAI2 peripheral clocks enable
Set and reset by software.
0: SAI2 peripheral clocks disabled (default after reset)
1: SAI2 peripheral clocks enabled
The peripheral clocks of the SAI2 are the kernel clock selected by SAI23SEL and provided to SAI_CK_A and SAI_CK_B inputs, and the `rcc_pclk2` bus interface clock.
Bit 22 **SAI1EN**: SAI1 peripheral clocks enable
Set and reset by software.
0: SAI1 peripheral clocks disabled (default after reset)
1: SAI1 peripheral clocks enabled:
The peripheral clocks of the SAI1 are: the kernel clock selected by SAI1SEL and provided to SAI_CK_A and SAI_CK_B inputs, and the `rcc_pclk2` bus interface clock.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **SPI5EN**: SPI5 peripheral clocks enable
Set and reset by software.
0: SPI5 peripheral clocks disabled (default after reset)
1: SPI5 peripheral clocks enabled:
The peripheral clocks of the SPI5 are the kernel clock selected by SPI45SEL and provided to `com_clk` input, and the `rcc_pclk2` bus interface clock.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM17EN**: TIM17 peripheral clock enable
Set and reset by software.
0: TIM17 peripheral clock disabled (default after reset)
1: TIM17 peripheral clock enabled

Bit 17 **TIM16EN**: TIM16 peripheral clock enable
Set and reset by software.
0: TIM16 peripheral clock disabled (default after reset)
1: TIM16 peripheral clock enabled

Bit 16 **TIM15EN**: TIM15 peripheral clock enable
Set and reset by software.
0: TIM15 peripheral clock disabled (default after reset)
1: TIM15 peripheral clock enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SPI4EN**: SPI4 Peripheral Clocks Enable
Set and reset by software.
0: SPI4 peripheral clocks disabled (default after reset)
1: SPI4 peripheral clocks enabled:
The peripheral clocks of the SPI4 are: the kernel clock selected by SPI45SEL and provided to `spi_ker_ck` input, and the `rcc_pclk2` bus interface clock.

Bit 12 **SPI1EN**: SPI1 Peripheral Clocks Enable
Set and reset by software.
0: SPI1 peripheral clocks disabled (default after reset)
1: SPI1 peripheral clocks enabled:
The peripheral clocks of the SPI1 are: the kernel clock selected by I2S123SRC and provided to `spi_ker_ck` input, and the `rcc_pclk2` bus interface clock.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10EN**: USART10 peripheral clocks enable
Set and reset by software.
0: USART10 peripheral clocks disabled (default after reset)
1: USART10 peripheral clocks enabled:
The peripheral clocks of the USART10 are the kernel clock selected by USART16910SEL and provided to UCKL input, and the `rcc_pclk2` bus interface clock.
Bit 6 **UART9EN**: UART9 peripheral clocks enable
Set and reset by software.
0: UART9 peripheral clocks disabled (default after reset)
1: UART9 peripheral clocks enabled:
The peripheral clocks of the UART9 are the kernel clock selected by USART16910SEL and provided to UCKL input, and the rcc_pclk2 bus interface clock.

Bit 5 **USART6EN**: USART6 peripheral clocks enable
Set and reset by software.
0: USART6 peripheral clocks disabled (default after reset)
1: USART6 peripheral clocks enabled:
The peripheral clocks of the USART6 are the kernel clock selected by USART16910SEL and provided to UCKL input, and the rcc_pclk2 bus interface clock.

Bit 4 **USART1EN**: USART1 peripheral clocks enable
Set and reset by software.
0: USART1 peripheral clocks disabled (default after reset)
1: USART1 peripheral clocks enabled:
The peripheral clocks of the USART1 are the kernel clock selected by USART16910SEL and provided to UCKL input, and the rcc_pclk2 bus interface clock.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8EN**: TIM8 peripheral clock enable
Set and reset by software.
0: TIM8 peripheral clock disabled (default after reset)
1: TIM8 peripheral clock enabled

Bit 0 **TIM1EN**: TIM1 peripheral clock enable
Set and reset by software.
0: TIM1 peripheral clock disabled (default after reset)
1: TIM1 peripheral clock enabled

### 8.7.46 RCC APB4 clock register (RCC_APB4ENR)

Address offset: 0x154
Reset value: 0x0001 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res.</td>
</tr>
<tr>
<td>30</td>
<td>Res.</td>
</tr>
<tr>
<td>29</td>
<td>Res.</td>
</tr>
<tr>
<td>28</td>
<td>Res.</td>
</tr>
<tr>
<td>27</td>
<td>Res.</td>
</tr>
<tr>
<td>26</td>
<td>Res.</td>
</tr>
<tr>
<td>25</td>
<td>DFSDMA2EN</td>
</tr>
<tr>
<td>24</td>
<td>Res.</td>
</tr>
<tr>
<td>23</td>
<td>DTSEN</td>
</tr>
<tr>
<td>22</td>
<td>Res.</td>
</tr>
<tr>
<td>21</td>
<td>Res.</td>
</tr>
<tr>
<td>20</td>
<td>Res.</td>
</tr>
<tr>
<td>19</td>
<td>Res.</td>
</tr>
<tr>
<td>18</td>
<td>Res.</td>
</tr>
<tr>
<td>17</td>
<td>Res.</td>
</tr>
<tr>
<td>16</td>
<td>RTCAPBEN</td>
</tr>
<tr>
<td>15</td>
<td>Res.</td>
</tr>
<tr>
<td>14</td>
<td>Res.</td>
</tr>
<tr>
<td>13</td>
<td>Res.</td>
</tr>
<tr>
<td>12</td>
<td>Res.</td>
</tr>
<tr>
<td>11</td>
<td>Res.</td>
</tr>
<tr>
<td>10</td>
<td>VREFEN</td>
</tr>
<tr>
<td>9</td>
<td>COMP12EN</td>
</tr>
<tr>
<td>8</td>
<td>DAC2EN</td>
</tr>
<tr>
<td>7</td>
<td>Res.</td>
</tr>
<tr>
<td>6</td>
<td>LPTIM3EN</td>
</tr>
<tr>
<td>5</td>
<td>LPTIM2EN</td>
</tr>
<tr>
<td>4</td>
<td>I2C4EN</td>
</tr>
<tr>
<td>3</td>
<td>SPI4EN</td>
</tr>
<tr>
<td>2</td>
<td>LPUART1EN</td>
</tr>
<tr>
<td>1</td>
<td>SYSCFGEN</td>
</tr>
<tr>
<td>0</td>
<td>Res.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:28  Reserved, must be kept at reset value.

Bit 27  **DFSDM2EN**: DFSDM2 peripheral clock enable
   Set and reset by software.
   0: DFSDM2 peripheral clock disabled (default after reset)
   1: DFSDM2 peripheral clock enabled

Bit 26  **DTSEN**: Digital temperature sensor peripheral clock enable
   Set and reset by software.
   0: DTS peripheral clock disabled (default after reset)
   1: DTS peripheral clock enabled

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

Bit 16  **RTCAPBEN**: RTC APB clock enable
   Set and reset by software.
   0: The register clock interface of the RTC (APB) is disabled
   1: The register clock interface of the RTC (APB) is enabled (default after reset)

Bit 15  **VREFEN**: VREF peripheral clock enable
   Set and reset by software.
   0: VREF peripheral clock disabled (default after reset)
   1: VREF peripheral clock enabled

Bit 14  **COMP12EN**: COMP1 and 2 peripheral clock enable
   Set and reset by software.
   0: COMP1 and 2 peripheral clock disabled (default after reset)
   1: COMP1 and 2 peripheral clock enabled

Bit 13  **DAC2EN**: DAC2 (containing one converter) peripheral clock enable
   Set and reset by software.
   0: DAC2 peripheral clock disabled (default after reset)
   1: DAC2 peripheral clock enabled

Bits 12:11  Reserved, must be kept at reset value.

Bit 10  **LPTIM3EN**: LPTIM3 peripheral clocks enable
   Set and reset by software.
   0: LPTIM3 peripheral clocks disabled (default after reset)
   1: LPTIM3 peripheral clocks enabled
   The peripheral clocks of the LPTIM3 are the kernel clock selected by LPTIM3SELD and provided to 
   **clk_lpt** input, and the **rcc_pclk4** bus interface clock.

Bit 9  **LPTIM2EN**: LPTIM2 peripheral clocks enable
   Set and reset by software.
   0: LPTIM2 peripheral clocks disabled (default after reset)
   1: LPTIM2 peripheral clocks enabled
   The peripheral clocks of the LPTIM2 are the kernel clock selected by LPTIM2SEL and provided to 
   **clk_lpt** input, and the **rcc_pclk4** bus interface clock.

Bit 8  Reserved, must be kept at reset value.

Bit 7  **I2C4EN**: I2C4 peripheral clocks enable
   Set and reset by software.
   0: I2C4 peripheral clocks disabled (default after reset)
   1: I2C4 peripheral clocks enabled
   The peripheral clocks of the I2C4 are the kernel clock selected by I2C4SEL and provided to I2CCLK 
   input, and the **rcc_pclk4** bus interface clock.
Bit 6  Reserved, must be kept at reset value.

Bit 5  **SPI6EN**: SPI6 peripheral clocks enable
       Set and reset by software.
       0: SPI6 peripheral clocks disabled (default after reset)
       1: SPI6 peripheral clocks enabled
       The peripheral clocks of the SPI6 are the kernel clock selected by SPI6SEL and provided to
       **com_clk** input, and the **rcc_pclk4** bus interface clock.

Bit 4  Reserved, must be kept at reset value.

Bit 3  **LPUART1EN**: LPUART1 peripheral clocks enable
       Set and reset by software.
       0: LPUART1 peripheral clocks disabled (default after reset)
       1: LPUART1 peripheral clocks enabled
       The peripheral clocks of the LPUART1 are the kernel clock selected by LPUART1SEL and provided
to UCLK input, and the **rcc_pclk4** bus interface clock.

Bit 2  Reserved, must be kept at reset value.

Bit 1  **SYSCFGEN**: SYSCFG peripheral clock enable
       Set and reset by software.
       0: SYSCFG peripheral clock disabled (default after reset)
       1: SYSCFG peripheral clock enabled

Bit 0  Reserved, must be kept at reset value.

### 8.7.47  **RCC AHB3 sleep clock register (RCC_AHB3LPENR)**

Address offset: 0x15C

Reset value: 0xFDE9 5131
Bit 31 **AXISRAM1LPEN**: AXISRAM1 block clock enable during CSleep mode
Set and reset by software.
0: AXISRAM1 interface clock disabled during CSleep mode
1: AXISRAM1 interface clock enabled during CSleep mode (default after reset)

Bit 30 **ITCMLPEN**: ITCM block clock enable during CSleep mode
Set and reset by software.
0: ITCM interface clock disabled during CSleep mode
1: ITCM interface clock enabled during CSleep mode (default after reset)

Bit 29 **DTCM2LPEN**: DTCM2 block clock enable during CSleep mode
Set and reset by software.
0: DTCM2 interface clock disabled during CSleep mode
1: DTCM2 interface clock enabled during CSleep mode (default after reset)

Bit 28 **DTCM1LPEN**: DTCM1 block clock enable during CSleep mode
Set and reset by software.
0: DTCM1 interface clock disabled during CSleep mode
1: DTCM1 interface clock enabled during CSleep mode (default after reset)

Bit 27 **AXISRAM3LPEN**: AXISRAM3 block clock enable during CSleep mode
Set and reset by software.
0: AXISRAM3 interface clock disabled during CSleep mode
1: AXISRAM3 interface clock enabled during CSleep mode (default after reset)

Bit 26 **AXISRAM2LPEN**: AXISRAM2 block clock enable during CSleep mode
Set and reset by software.
0: AXISRAM2 interface clock disabled during CSleep mode
1: AXISRAM2 interface clock enabled during CSleep mode (default after reset)

Bit 25 Reserved, must be kept at reset value.

Bit 24 **GFXMULPEN**: GFXMMU block clock enable during CSleep mode
Set and reset by software.
0: GFXMMU interface clock disabled during CSleep mode
1: GFXMMU interface clock enabled during CSleep mode (default after reset)

Bit 23 **OTFD2LPEN**: OTFD2 block clock enable during CSleep mode
Set and reset by software.
0: OTFD2 interface clock disabled during CSleep mode
1: OTFD2 interface clock enabled during CSleep mode (default after reset)

Bit 22 **OTFD1LPEN**: OTFD1 block clock enable during CSleep mode
Set and reset by software.
0: OTFD1 interface clock disabled during CSleep mode
1: OTFD1 interface clock enabled during CSleep mode (default after reset)

Bit 21 **IOMNGRLPEN**: IOMNGR block clock enable during CSleep mode
Set and reset by software.
0: IOMNGR interface clock disabled during CSleep mode
1: IOMNGR interface clock enabled during CSleep mode (default after reset)

Bit 20 Reserved, must be kept at reset value.

Bit 19 **OCTOSPI2LPEN**: OCTOSPI2 and OCTOSPI2 delay clock enable during CSleep mode
Set and reset by software.
0: OCTOSPI2 and OCTOSPI2 delay clock disabled during CSleep mode
1: OCTOSPI2 and OCTOSPI2 delay clock enabled during CSleep mode (default after reset)
Bits 18:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC1LPEN**: SDMMC1 and SDMMC1 delay clock enable during CSleep mode  
Set and reset by software.  
0: SDMMC1 and SDMMC1 delay clock disabled during CSleep mode  
1: SDMMC1 and SDMMC1 delay clock enabled during CSleep mode (default after reset)

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OCTOSPI1LPEN**: OCTOSPI1 and OCTOSPI1 delay clock enable during CSleep mode  
Set and reset by software.  
0: OCTOSPI1 and OCTOSPI1 delay clock disabled during CSleep mode  
1: OCTOSPI1 and OCTOSPI1 delay clock enabled during CSleep mode (default after reset)

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCLPEN**: FMC peripheral clocks enable during CSleep mode  
Set and reset by software.  
0: FMC peripheral clocks disabled during CSleep mode  
1: FMC peripheral clocks enabled during CSleep mode (default after reset)  
The peripheral clocks of the FMC are the kernel clock selected by FMCSEL and provided to KCK_FMC input, and the rcc_hclk3 bus interface clock.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLITFLPEN**: FLITF clock enable during CSleep mode  
Set and reset by software.  
0: FLITF clock disabled during CSleep mode  
1: FLITF clock enabled during CSleep mode (default after reset)

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **JPQDECLPEN**: JPGDEC clock enable during CSleep mode  
Set and reset by software.  
0: JPGDEC peripheral clock disabled during CSleep mode  
1: JPGDEC peripheral clock enabled during CSleep mode (default after reset)

Bit 4 **DMA2DLPEN**: DMA2D clock enable during CSleep mode  
Set and reset by software.  
0: DMA2D peripheral clock disabled during CSleep mode  
1: DMA2D peripheral clock enabled during CSleep mode (default after reset)

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **MDMALPEN**: MDMA clock enable during CSleep mode  
Set and reset by software.  
0: MDMA peripheral clock disabled during CSleep mode  
1: MDMA peripheral clock enabled during CSleep mode (default after reset)
8.7.48  **RCC AHB1 sleep clock register (RCC_AHB1LPENR)**

Address offset: 0x160  
Reset value: 0x0600 0223

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>USB1OTGLPEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 26 **USB1ULPILPEN**: USB_PHY1 clock enable during CSleep mode  
Set and reset by software.  
0: USB_PHY1 peripheral clock disabled during CSleep mode  
1: USB_PHY1 peripheral clock enabled during CSleep mode (default after reset)

Bit 25 **USB1OTGLPEN**: USB1OTG peripheral clock enable during CSleep mode  
Set and reset by software.  
0: USB1OTG peripheral clock disabled during CSleep mode  
1: USB1OTG peripheral clock enabled during CSleep mode (default after reset)  
The peripheral clocks of the USB1OTG are the kernel clock selected by USBSEL and the `rcc_hclk1` bus interface clock.

Bits 24:10  Reserved, must be kept at reset value.

Bit 9 **CRCILPEN**: CRC peripheral clock enable during CSleep mode  
Set and reset by software.  
0: CRC peripheral clock disabled during CSleep mode  
1: CRC peripheral clock enabled during CSleep mode (default after reset)

Bits 8:6  Reserved, must be kept at reset value.

Bit 5 **ADC12LPEN**: ADC1 and 2 peripheral clocks enable during CSleep mode  
Set and reset by software.  
0: ADC1 and 2 peripheral clocks disabled during CSleep mode  
1: ADC1 and 2 peripheral clocks enabled during CSleep mode (default after reset)  
The peripheral clocks of the ADC1 and 2 are the kernel clock selected by ADCSEL and provided to ADCx_CK input, and the `rcc_hclk1` bus interface clock.
Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **DMA2LPEN**: DMA2 clock enable during CSleep mode
Set and reset by software.
0: DMA2 clock disabled during CSleep mode
1: DMA2 clock enabled during CSleep mode (default after reset)

Bit 0 **DMA1LPEN**: DMA1 clock enable during CSleep mode
Set and reset by software.
0: DMA1 clock disabled during CSleep mode
1: DMA1 clock enabled during CSleep mode (default after reset)

### 8.7.49 RCC AHB2 sleep clock register (RCC_AHB2LPENR)

Address offset: 0x164

Reset value: 0x6000 0A71

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **AHBSRAM2LPEN**: AHBSRAM2 clock enable during CSleep mode
Set and reset by software.
0: AHBSRAM2 clock disabled during CSleep mode
1: AHBSRAM2 clock enabled during CSleep mode (default after reset)

Bit 29 **AHBSRAM1LPEN**: AHBSRAM1 clock enable during CSleep mode
Set and reset by software.
0: AHBSRAM1 clock disabled during CSleep mode
1: AHBSRAM1 clock enabled during CSleep mode (default after reset)

Bits 28:12 Reserved, must be kept at reset value.

Bit 11 **DFSDM0DMALPEN**: DFSDM0DMA clock enable during CSleep mode
Set and reset by software.
0: DFSDM0DMA clock disabled during CSleep mode
1: DFSDM0DMA clock enabled during CSleep mode (default after reset)

Bit 10 Reserved, must be kept at reset value.
8.7.50 RCC AHB4 sleep clock register (RCC_AHB4LPENR)

Address offset: 0x168

Reset value: 0x3020 07FF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Bits 31:30  Reserved, must be kept at reset value.

Bit 29  **SRDSRAMLPE**: SmartRun domain SRAM clock enable during CSleep mode
Set and reset by software.
0: SRDSRAM clock disabled during CSleep mode
1: SRDSRAM clock enabled during CSleep mode (default after reset)

Bit 28  **BKPRAMLPE**: Backup RAM clock enable during CSleep mode
Set and reset by software.
0: Backup RAM clock disabled during CSleep mode
1: Backup RAM clock enabled during CSleep mode (default after reset)

Bits 27:22  Reserved, must be kept at reset value.

Bit 21  **BDMA2LPEN**: SmartRun domain DMA and DMAMUX clock enable during CSleep mode
Set and reset by software.
0: BDMA2 and DMAMUX2 clock disabled during CSleep mode
1: BDMA2 and DMAMUX2 clock enabled during CSleep mode (default after reset)

Bits 20:11  Reserved, must be kept at reset value.

Bit 10  **GPIOKLPEN**: GPIOK peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOK peripheral clock disabled during CSleep mode
1: GPIOK peripheral clock enabled during CSleep mode (default after reset)

Bit 9  **GPIOJLPEN**: GPIOJ peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOJ peripheral clock disabled during CSleep mode
1: GPIOJ peripheral clock enabled during CSleep mode (default after reset)

Bit 8  **GPIOLPEN**: GPIOI peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOI peripheral clock disabled during CSleep mode
1: GPIOI peripheral clock enabled during CSleep mode (default after reset)

Bit 7  **GPIOHLPE**: GPIOH peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOH peripheral clock disabled during CSleep mode
1: GPIOH peripheral clock enabled during CSleep mode (default after reset)

Bit 6  **GPIOGLPE**: GPIOG peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOG peripheral clock disabled during CSleep mode
1: GPIOG peripheral clock enabled during CSleep mode (default after reset)

Bit 5  **GPIOFLPE**: GPIOF peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOF peripheral clock disabled during CSleep mode
1: GPIOF peripheral clock enabled during CSleep mode (default after reset)

Bit 4  **GPIOELPE**: GPIOE peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOE peripheral clock disabled during CSleep mode
1: GPIOE peripheral clock enabled during CSleep mode (default after reset)
Bit 3 GPIODL PEN: GPIOD peripheral clock enable during CSleep mode  
Set and reset by software.  
0: GPIOD peripheral clock disabled during CSleep mode  
1: GPIOD peripheral clock enabled during CSleep mode (default after reset)

Bit 2 GPIOCL PEN: GPIOC peripheral clock enable during CSleep mode  
Set and reset by software.  
0: GPIOC peripheral clock disabled during CSleep mode  
1: GPIOC peripheral clock enabled during CSleep mode (default after reset)

Bit 1 GPIOBL PEN: GPIOB peripheral clock enable during CSleep mode  
Set and reset by software.  
0: GPIOB peripheral clock disabled during CSleep mode  
1: GPIOB peripheral clock enabled during CSleep mode (default after reset)

Bit 0 GPIOAL PEN: GPIOA peripheral clock enable during CSleep mode  
Set and reset by software.  
0: GPIOA peripheral clock disabled during CSleep mode  
1: GPIOA peripheral clock enabled during CSleep mode (default after reset)

### 8.7.51 RCC APB3 sleep clock register (RCC_APB3LPENR)

Address offset: 0x16C  
Reset value: 0x0000 0048

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 WW DGL PEN: WWDG clock enable during CSleep mode  
Set and reset by software.  
0: WWDG clock disable during CSleep mode  
1: WWDG clock enabled during CSleep mode (default after reset)

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 LTDC LPEN: LTDC peripheral clock enable during CSleep mode  
Provides the pixel clock (/lcd_clk) to the LTDC block.  
Set and reset by software.  
0: LTDC clock disabled during CSleep mode  
1: LTDC clock provided to the LTDC during CSleep mode (default after reset)

Bits 2:0 Reserved, must be kept at reset value.
8.7.52 RCC APB1 low-sleep clock register (RCC_APB1LLPENR)

Address offset: 0x170
Reset value: 0xE8FF C3FF

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Default</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>UART8LPEN</td>
<td>rw</td>
<td>0</td>
<td>Set and reset by software. 0: UART8 peripheral clocks disabled during CSleep mode 1: UART8 peripheral clocks enabled during CSleep mode (default after reset): The peripheral clocks of the UART8 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the rcc_pclk1 bus interface clock.</td>
</tr>
<tr>
<td>30</td>
<td>UART7LPEN</td>
<td>rw</td>
<td>0</td>
<td>Set and reset by software. 0: UART7 peripheral clocks disabled during CSleep mode 1: UART7 peripheral clocks enabled during CSleep mode (default after reset): The peripheral clocks of the UART7 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the rcc_pclk1 bus interface clock.</td>
</tr>
<tr>
<td>29</td>
<td>DAC1LPEN</td>
<td>rw</td>
<td>0</td>
<td>DAC1 (containing two converters) peripheral clock enable during CSleep mode Set and reset by software. 0: DAC1 peripheral clock disabled during CSleep mode 1: DAC1 peripheral clock enabled during CSleep mode (default after reset)</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>rw</td>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>HDMICECLPEN</td>
<td>rw</td>
<td>0</td>
<td>HDMI-CEC peripheral clocks enable during CSleep mode Set and reset by software. 0: HDMI-CEC peripheral clocks disabled during CSleep mode 1: HDMI-CEC peripheral clocks enabled during CSleep mode (default after reset): The peripheral clocks of the HDMI-CEC are the kernel clock selected by CECSEL and provided to clk_cec input, and the rcc_pclk1 bus interface clock.</td>
</tr>
<tr>
<td>26-24</td>
<td>Reserved</td>
<td>rw</td>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23</td>
<td>I2C3LPEN</td>
<td>rw</td>
<td>0</td>
<td>I2C3 peripheral clocks enable during CSleep mode Set and reset by software. 0: I2C3 peripheral clocks disabled during CSleep mode 1: I2C3 peripheral clocks enabled during CSleep mode (default after reset): The peripheral clocks of the I2C3 are the kernel clock selected by I2C123SEL and provided to I2CCLK input, and the rcc_pclk1 bus interface clock.</td>
</tr>
</tbody>
</table>
Bit 22  **I2C2LPEN**: I2C2 peripheral clocks enable during CSleep mode  
Set and reset by software.  
0: I2C2 peripheral clocks disabled during CSleep mode  
1: I2C2 peripheral clocks enabled during CSleep mode (default after reset):  
The peripheral clocks of the I2C2 are the kernel clock selected by I2C123SEL and provided to I2CCCLK input, and the `rcc_pclk1` bus interface clock.

Bit 21  **I2C1LPEN**: I2C1 peripheral clocks enable during CSleep mode  
Set and reset by software.  
0: I2C1 peripheral clocks disabled during CSleep mode  
1: I2C1 peripheral clocks enabled during CSleep mode (default after reset):  
The peripheral clocks of the I2C1 are the kernel clock selected by I2C123SEL and provided to I2CCCLK input, and the `rcc_pclk1` bus interface clock.

Bit 20  **UART5LPEN**: UART5 peripheral clocks enable during CSleep mode  
Set and reset by software.  
0: UART5 peripheral clocks disabled during CSleep mode  
1: UART5 peripheral clocks enabled during CSleep mode (default after reset):  
The peripheral clocks of the UART5 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the `rcc_pclk1` bus interface clock.

Bit 19  **UART4LPEN**: UART4 peripheral clocks enable during CSleep mode  
Set and reset by software.  
0: UART4 peripheral clocks disabled during CSleep mode  
1: UART4 peripheral clocks enabled during CSleep mode (default after reset):  
The peripheral clocks of the UART4 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the `rcc_pclk1` bus interface clock.

Bit 18  **USART3LPEN**: USART3 peripheral clocks enable during CSleep mode  
Set and reset by software.  
0: USART3 peripheral clocks disabled during CSleep mode  
1: USART3 peripheral clocks enabled during CSleep mode (default after reset):  
The peripheral clocks of the USART3 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the `rcc_pclk1` bus interface clock.

Bit 17  **USART2LPEN**: USART2 peripheral clocks enable during CSleep mode  
Set and reset by software.  
0: USART2 peripheral clocks disabled during CSleep mode  
1: USART2 peripheral clocks enabled during CSleep mode (default after reset):  
The peripheral clocks of the USART2 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the `rcc_pclk1` bus interface clock.

Bit 16  **SPDIFRXLPEN**: SPDIFRX peripheral clocks enable during CSleep mode  
Set and reset by software.  
0: SPDIFRX peripheral clocks disabled during CSleep mode  
1: SPDIFRX peripheral clocks enabled during CSleep mode (default after reset):  
The peripheral clocks of the SPDIFRX are: the kernel clock selected by SPDIFRXSEL and provided to SPDIFRX_CLK input, and the `rcc_pclk1` bus interface clock.

Bit 15  **SPI3LPEN**: SPI3 peripheral clocks enable during CSleep mode  
Set and reset by software.  
0: SPI3 peripheral clocks disabled during CSleep mode  
1: SPI3 peripheral clocks enabled during CSleep mode (default after reset):  
The peripheral clocks of the SPI3 are the kernel clock selected by I2S123SRC and provided to com_clk input, and the `rcc_pclk1` bus interface clock.
Bit 14 SPI2LPEN: SPI2 peripheral clocks enable during CSleep mode
Set and reset by software.
0: SPI2 peripheral clocks disabled during CSleep mode
1: SPI2 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SPI2 are the kernel clock selected by I2S123SRC and provided to com_clk input, and the rcc_pclk1 bus interface clock.

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 LPTIM1LPEN: LPTIM1 peripheral clocks enable during CSleep mode
Set and reset by software.
0: LPTIM1 peripheral clocks disabled during CSleep mode
1: LPTIM1 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the LPTIM1 are the kernel clock selected by LPTIM1SEL and provided to clk_lpt input, and the rcc_pclk1 bus interface clock.

Bit 8 TIM14LPEN: TIM14 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM14 peripheral clock disabled during CSleep mode
1: TIM14 peripheral clock enabled during CSleep mode (default after reset)

Bit 7 TIM13LPEN: TIM13 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM13 peripheral clock disabled during CSleep mode
1: TIM13 peripheral clock enabled during CSleep mode (default after reset)

Bit 6 TIM12LPEN: TIM12 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM12 peripheral clock disabled during CSleep mode
1: TIM12 peripheral clock enabled during CSleep mode (default after reset)

Bit 5 TIM7LPEN: TIM7 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM7 peripheral clock disabled during CSleep mode
1: TIM7 peripheral clock enabled during CSleep mode (default after reset)

Bit 4 TIM6LPEN: TIM6 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM6 peripheral clock disabled during CSleep mode
1: TIM6 peripheral clock enabled during CSleep mode (default after reset)

Bit 3 TIM5LPEN: TIM5 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM5 peripheral clock disabled during CSleep mode
1: TIM5 peripheral clock enabled during CSleep mode (default after reset)
Bit 2 **TIM4LPEN**: TIM4 peripheral clock enable during CSleep mode  
Set and reset by software.  
0: TIM4 peripheral clock disabled during CSleep mode  
1: TIM4 peripheral clock enabled during CSleep mode (default after reset)

Bit 1 **TIM3LPEN**: TIM3 peripheral clock enable during CSleep mode  
Set and reset by software.  
0: TIM3 peripheral clock disabled during CSleep mode  
1: TIM3 peripheral clock enabled during CSleep mode (default after reset)

Bit 0 **TIM2LPEN**: TIM2 peripheral clock enable during CSleep mode  
Set and reset by software.  
0: TIM2 peripheral clock disabled during CSleep mode  
1: TIM2 peripheral clock enabled during CSleep mode (default after reset)

### 8.7.53 RCC APB1 high-sleep clock register (RCC_APB1HLPNR)

Address offset: 0x174  
Reset value: 0x0000 0136

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **FDCANLPEN**: FDCAN peripheral clocks enable during CSleep mode  
Set and reset by software.  
0: FDCAN peripheral clocks disabled during CSleep mode  
1: FDCAN peripheral clocks enabled during CSleep mode (default after reset)  
The peripheral clocks of the FDCAN are: the kernel clock selected by FDCANSEL and provided to fdcan_clk input, and the rcc_pclk1 bus interface clock.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSLPEN**: MDIOS peripheral clock enable during CSleep mode  
Set and reset by software.  
0: MDIOS peripheral clock disabled during CSleep mode  
1: MDIOS peripheral clock enabled during CSleep mode (default after reset)

Bit 4 **OPAMPLPEN**: OPAMP peripheral clock enable during CSleep mode  
Set and reset by software.  
0: OPAMP peripheral clock disabled during CSleep mode  
1: OPAMP peripheral clock enabled during CSleep mode (default after reset)

Bit 3 Reserved, must be kept at reset value.
Bit 2 **SWPLPEN**: SWP peripheral clocks enable during CSleep mode
Set and reset by software.
0: SWP peripheral clocks disabled during CSleep mode
1: SWP peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SWP are the kernel clock selected by SWPSEL and provided to SWPCLK input, and the \textit{rcc\_pclk1} bus interface clock.

Bit 1 **CRSLPEN**: clock recovery system peripheral clock enable during CSleep mode
Set and reset by software.
0: CRS peripheral clock disabled during CSleep mode
1: CRS peripheral clock enabled during CSleep mode (default after reset)

Bit 0 Reserved, must be kept at reset value.

8.7.54 **RCC APB2 sleep clock register (RCC\_APB2LPENR)**
Address offset: 0x178
Reset value: 0x40D7 30F3

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DFSDM1LPEN**: DFSDM1 peripheral clocks enable during CSleep mode
Set and reset by software.
0: DFSDM1 peripheral clocks disabled during CSleep mode
1: DFSDM1 peripheral clocks enabled during CSleep mode (default after reset)
DFSDM1 peripheral clocks are the kernel clocks selected by SAI1SEL and DFSDM1SEL and provided to 
A\textit{clk} and 
\textit{clk} inputs respectively, and the \textit{rcc\_pclk2} bus interface clock.

Bits 29:24 Reserved, must be kept at reset value.

Bit 23 **SAI2LPEN**: SAI2 peripheral clocks enable during CSleep mode
Set and reset by software.
0: SAI2 peripheral clocks disabled during CSleep mode
1: SAI2 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SAI2 are the kernel clock selected by SAI23SEL and provided to
\textit{SAI\_CK\_A} and \textit{SAI\_CK\_B} inputs, and the \textit{rcc\_pclk2} bus interface clock.
Bit 22 **SAI1LPEN**: SAI1 peripheral clocks enable during CSleep mode
Set and reset by software.
0: SAI1 peripheral clocks disabled during CSleep mode
1: SAI1 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SAI1 are: the kernel clock selected by SAI1SEL and provided to SAI_CK_A and SAI_CK_B inputs, and the \texttt{rcc\_pclk2} bus interface clock.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **SPI5LPEN**: SPI5 peripheral clocks enable during CSleep mode
Set and reset by software.
0: SPI5 peripheral clocks disabled during CSleep mode
1: SPI5 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SPI5 are the kernel clock selected by SPI45SEL and provided to \texttt{com\_clk} input, and the \texttt{rcc\_pclk2} bus interface clock.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM17LPEN**: TIM17 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM17 peripheral clock disabled during CSleep mode
1: TIM17 peripheral clock enabled during CSleep mode (default after reset)

Bit 17 **TIM16LPEN**: TIM16 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM16 peripheral clock disabled during CSleep mode
1: TIM16 peripheral clock enabled during CSleep mode (default after reset)

Bit 16 **TIM15LPEN**: TIM15 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM15 peripheral clock disabled during CSleep mode
1: TIM15 peripheral clock enabled during CSleep mode (default after reset)

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SPI4LPEN**: SPI4 peripheral clock enable during CSleep mode
Set and reset by software.
0: SPI4 peripheral clocks disabled during CSleep mode
1: SPI4 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SPI4 are: the kernel clock selected by SPI45SEL and provided to \texttt{com\_clk} input, and the \texttt{rcc\_pclk2} bus interface clock.

Bit 12 **SPI1LPEN**: SPI1 peripheral clock enable during CSleep mode
Set and reset by software.
0: SPI1 peripheral clocks disabled during CSleep mode
1: SPI1 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SPI1 are: the kernel clock selected by I2S123SRC and provided to \texttt{com\_clk} input, and the \texttt{rcc\_pclk2} bus interface clock.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10LPEN**: USART10 peripheral clock enable during CSleep mode
Set and reset by software.
0: USART10 peripheral clocks disabled during CSleep mode
1: USART10 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the USART10 are the kernel clock selected by USART16910SEL and provided to UCLK input, and the \texttt{rcc\_pclk2} bus interface clock.
8.7.55 **RCC APB4 sleep clock register (RCC_APB4LPENR)**

Address offset: 0x17C

Reset value: 0x0C01 E6AA

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

Bit 6 **UART9LPEN**: UART9 peripheral clock enable during CSleep mode

0: UART9 peripheral clocks disabled during CSleep mode
1: UART9 peripheral clocks enabled during CSleep mode (default after reset)

The peripheral clocks of the UART9 are the kernel clock selected by USART16910SEL and provided to UCLK input, and the **rcc_pclk2** bus interface clock.

Bit 5 **USART6LPEN**: USART6 peripheral clock enable during CSleep mode

0: USART6 peripheral clocks disabled during CSleep mode
1: USART6 peripheral clocks enabled during CSleep mode (default after reset)

The peripheral clocks of the USART6 are the kernel clock selected by USART16910SEL and provided to UCLK input, and the **rcc_pclk2** bus interface clock.

Bit 4 **USART1LPEN**: USART1 peripheral clock enable during CSleep mode

0: USART1 peripheral clocks disabled during CSleep mode
1: USART1 peripheral clocks enabled during CSleep mode (default after reset)

The peripheral clocks of the USART1 are the kernel clock selected by USART16910SEL and provided to UCLK inputs, and the **rcc_pclk2** bus interface clock.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8LPEN**: TIM8 peripheral clock enable during CSleep mode

0: TIM8 peripheral clock disabled during CSleep mode
1: TIM8 peripheral clock enabled during CSleep mode (default after reset)

Bit 0 **TIM1LPEN**: TIM1 peripheral clock enable during CSleep mode

0: TIM1 peripheral clock disabled during CSleep mode
1: TIM1 peripheral clock enabled during CSleep mode (default after reset)
Bits 31:28  Reserved, must be kept at reset value.

Bit 27  **DFSDM2LPEN:** DFSDM2 peripheral clock enable during CSleep mode
Set and reset by software.
0: DFSDM2 peripheral clock disabled during CSleep mode
1: DFSDM2 peripheral clock enabled during CSleep mode (default after reset)

Bit 26  **DTS LPEN:** temperature sensor peripheral clock enable during CSleep mode
Set and reset by software.
0: DTS peripheral clock disabled during CSleep mode
1: DTS peripheral clock enabled during CSleep mode (default after reset)

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

Bit 16  **RTCAPBLPEN:** RTC APB clock enable during CSleep mode
Set and reset by software.
0: The register clock interface of the RTC (APB) is disabled during CSleep mode
1: The register clock interface of the RTC (APB) is enabled during CSleep mode (default after reset)

Bit 15  **VREFLPEN:** VREF peripheral clock enable during CSleep mode
Set and reset by software.
0: VREF peripheral clock disabled during CSleep mode
1: VREF peripheral clock enabled during CSleep mode (default after reset)

Bit 14  **COMP12LPEN:** COMP1 and 2 peripheral clock enable during CSleep mode
Set and reset by software.
0: COMP1 and 2 peripheral clock disabled during CSleep mode
1: COMP1 and 2 peripheral clock enabled during CSleep mode (default after reset)

Bit 13  **DAC2LPEN:** DAC2 (containing one converter) peripheral clock enable during CSleep mode
Set and reset by software.
0: DAC2 peripheral clock disabled during CSleep mode
1: DAC2 peripheral clock enabled during CSleep mode (default after reset)

Bits 12:11  Reserved, must be kept at reset value.

Bit 10  **LPTIM3LPEN:** LPTIM3 peripheral clocks enable during CSleep mode
Set and reset by software.
0: LPTIM3 peripheral clocks disabled during CSleep mode
1: LPTIM3 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the LPTIM3 are the kernel clock selected by LPTIM345SEL and provided to clk_lpt input, and the rcc_pclk4 bus interface clock.

Bit 9  **LPTIM2LPEN:** LPTIM2 peripheral clocks enable during CSleep mode
Set and reset by software.
0: LPTIM2 peripheral clocks disabled during CSleep mode
1: LPTIM2 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the LPTIM2 are the kernel clock selected by LPTIM2SEL and provided to clk_lpt input, and the rcc_pclk4 bus interface clock.

Bit 8  Reserved, must be kept at reset value.

Bit 7  **I2C4LPEN:** I2C4 peripheral clocks enable during CSleep mode
Set and reset by software.
0: I2C4 peripheral clocks disabled during CSleep mode
1: I2C4 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the I2C4 are the kernel clock selected by I2C4SEL and provided to I2CCLK input, and the rcc_pclk4 bus interface clock.
Bit 6  Reserved, must be kept at reset value.

Bit 5  **SPI6LPEN**: SPI6 peripheral clocks enable during CSleep mode
Set and reset by software.
0: SPI6 peripheral clocks disabled during CSleep mode
1: SPI6 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SPI6 are the kernel clock selected by SPI6SEL and provided to `com_ck` input, and the `rcc_pclk4` bus interface clock.

Bit 4  Reserved, must be kept at reset value.

Bit 3  **LPUART1LPEN**: LPUART1 peripheral clocks enable during CSleep mode
Set and reset by software.
0: LPUART1 peripheral clocks disabled during CSleep mode
1: LPUART1 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the LPUART1 are the kernel clock selected by LPUART1SEL and provided to UCLK input, and the `rcc_pclk4` bus interface clock.

Bit 2  Reserved, must be kept at reset value.

Bit 1  **SYSCFGLPEN**: SYSCFG peripheral clock enable during CSleep mode
Set and reset by software.
0: SYSCFG peripheral clock disabled during CSleep mode
1: SYSCFG peripheral clock enabled during CSleep mode (default after reset)

Bit 0  Reserved, must be kept at reset value.

### 8.8 RCC register map

#### Table 63. RCC register map and reset values

| Offset | Register name      | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | RCC_CR             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x004  | RCC_HSICFGR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | RCC_CRRCR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00C  | RCC_CSICFGR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x010  | RCC_CFFGR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x014  | Reserved           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Reset value: **000000 000000000 000 000101**

Reset value: **10000-- --------**

Reset value: **----------**
Table 63. RCC register map and reset values (continued)

| Offset | Register name          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x018  | RCC_CDCFG1R1           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x01C  | RCC_CDCFG2R2           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x020  | RCC_SRDCFG1R           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x024  | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x028  | RCC_PLLCKSELR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x02C  | RCC_PLLCFG1R           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x030  | RCC_PLL1DIVR           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x034  | RCC_PLL1FRACR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x038  | RCC_PLL2DIVR           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x03C  | RCC_PLL2FRACR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x040  | RCC_PLL3DIVR           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x044  | RCC_PLL3FRACR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x048  | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04C  | RCC_CDCCIPR            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Reset value for each register is shown in hexadecimal format.
### Table 63. RCC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset Register name</th>
<th>Offset Register name</th>
<th>Offset Register name</th>
<th>Offset Register name</th>
<th>Offset Register name</th>
<th>Offset Register name</th>
<th>Offset Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x050</td>
<td>RCC_CDCCIP1R</td>
<td>SWPSEL</td>
<td>DF3MRSEL(1:0)</td>
<td>DF2MRSEL(1:0)</td>
<td>DF1MRSEL(1:0)</td>
<td>DF0MRSEL(1:0)</td>
<td>DFSEL(1:0)</td>
<td>DFSEL(1:0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x054</td>
<td>RCC_CDCCIP2R</td>
<td>LPTIMSEL(2:0)</td>
<td>USBSEL(1:0)</td>
<td>CECCSEL(1:0)</td>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x058</td>
<td>RCC_SRDCICPR</td>
<td>SPI6SEL(2:0)</td>
<td>DFSDM2SEL</td>
<td>Res.</td>
<td>0x058</td>
<td>RCC_CIER</td>
<td>Reset value</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x05C</td>
<td>Reserved</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x060</td>
<td>RCC_CIER</td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x064</td>
<td>RCC_CIFR</td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x068</td>
<td>RCC_CICR</td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x06C</td>
<td>Reserved</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x070</td>
<td>RCC_BDCR</td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x074</td>
<td>RCC_CSR</td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x078</td>
<td>Reserved</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x07C</td>
<td>RCC_AHB3RSTR</td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

Reset value 0x00000000.
### Table 63. RCC register map and reset values (continued)

| Offset  | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x080   | RCC_AHB1RSTR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x084   | RCC_AHB2RSTR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x088   | RCC_AHB4RSTR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08C   | RCC_APB3RSTR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x090   | RCC_APB1LRSTR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x094   | RCC_APB1HRSTR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x098   | RCC_APB2RSTR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x09C   | RCC_APB4RSTR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0A4   | RCC_SRDAMR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0A8   | RCC_SDAMR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0AC   | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**Note:** The table continues with more registers and reset values, but the above entries represent the format and structure of the table. The reset values are given for each register and are expressed in hexadecimal format. The table spans across multiple pages, indicating the continuation of the register map and reset values.
### Table 63. RCC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset value</th>
<th>Reset value</th>
<th>Offset value</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0B0</td>
<td>RCC_CKGAENR</td>
<td>JTAGCK</td>
<td>00000000</td>
<td>EXTCKG</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXRAMCKG</td>
<td>00000000</td>
<td>ECCRAMCKG</td>
<td>00000000</td>
</tr>
<tr>
<td>0x0B4</td>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>0x130</td>
<td>RCC_RSR</td>
<td>LPWRRSTF</td>
<td>00000000</td>
<td>WWORSTF</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WWORSTF</td>
<td>00000000</td>
<td>WWORSTF</td>
<td>00000000</td>
</tr>
<tr>
<td>0x134</td>
<td>RCC_AHB3ENR</td>
<td></td>
<td>00000000</td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>0x138</td>
<td>RCC_AHB1ENR</td>
<td></td>
<td>00000000</td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>0x140</td>
<td>RCC_AHB4ENR</td>
<td></td>
<td>00000000</td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>0x144</td>
<td>RCC_APB3ENR</td>
<td></td>
<td>00000000</td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>0x148</td>
<td>RCC_APB1LENR</td>
<td></td>
<td>00000000</td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>0x14C</td>
<td>RCC_APB1HENR</td>
<td></td>
<td>00000000</td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>0x150</td>
<td>RCC_APB2ENR</td>
<td></td>
<td>00000000</td>
<td></td>
<td>00000000</td>
</tr>
</tbody>
</table>

Note: Table continues with additional registers and reset values.
### Table 63. RCC register map and reset values (continued)

| Offset | Register name | Address | Bits | Reset value
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x154</td>
<td>RCC_APB4ENR</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x158</td>
<td>RCC_AHB1LPENR</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x15C</td>
<td>RCC_AHB2LPENR</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x160</td>
<td>RCC_AHB3LPENR</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x164</td>
<td>RCC_AHB4LPENR</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x168</td>
<td>RCC_AHB1LPENR</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x16C</td>
<td>RCC_APB2LPENR</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x170</td>
<td>RCC_APB1LPENR</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x174</td>
<td>RCC_APB1LPENR</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x178</td>
<td>RCC_APB2LPENR</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Offset**: Address space of the register

**Register name**: Name of the register

**Address**: Physical address of the register

**Bits**: Bit positions in the register

**Reset value**: Initial value of the register after reset
Table 63. RCC register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x17C  | RCC_APB4LPENR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 1  | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x120 to 0x1FC | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.3 on page 128 for the register boundary addresses.
9  Clock recovery system (CRS)

9.1  Introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means for oscillator output frequency evaluation, based on comparison with a selectable synchronization signal. It is capable of doing automatic adjustment of oscillator trimming based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In such case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, which is sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from the LSE oscillator output or it can be generated by user software.

9.2  CRS main features

- Selectable synchronization source with programmable prescaler and polarity:
  - External pin
  - LSE oscillator output
  - USB1 SOF packet reception
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster start-up convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCKOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

9.3  CRS implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>CRS1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIM width</td>
<td>6 bits</td>
</tr>
</tbody>
</table>
9.4 CRS functional description

9.4.1 CRS block diagram

Figure 66. CRS block diagram

Table 65 gives the list of CRS internal signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crs_it</td>
<td>Digital output</td>
<td>CRS interrupt</td>
</tr>
<tr>
<td>crs_pclk</td>
<td>Digital input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>hsi48_ck</td>
<td>Digital input</td>
<td>HSI48 oscillator clock</td>
</tr>
</tbody>
</table>
9.5.1 Synchronization input

The CRS synchronization (SYNC) source, selectable through the CRS_CFGR register, can be the signal from an external signal (SYNC), the LSE clock, or the OTG HS SOF signal. This source signal also has a configurable polarity and can then be divided by a programmable binary prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

For more information on the CRS synchronization source configuration, refer to Section 9.8.2: CRS configuration register (CRS_CFGR).

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

9.5.2 Frequency error measurement

The frequency error counter is a 16-bit down/up counter which is reloaded with the RELOAD value on each SYNC event. It starts counting down till it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit where it eventually stops (if no SYNC event is received) and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the downcounting phase (before reaching the zero value), it means that the actual frequency is lower than the target (and so, that the TRIM value must be incremented), while when it is detected during the upcounting phase it means that the actual frequency is higher (and that the TRIM value must be decremented).

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crs_trim[0:5]</td>
<td>Digital output</td>
<td>HSI48 oscillator smooth trimming value</td>
</tr>
<tr>
<td>crs_sync0, crs_sync1, crs_sync2</td>
<td>Digital input</td>
<td>SYNC signal source selection (SYNC, LSE or OTG_HS)</td>
</tr>
</tbody>
</table>
Figure 67. CRS counter behavior

- **CRS counter value**
- **RELOAD**
- **OUTRANGE** (128 x FELIM)
- **WARNING LIMIT** (3 x FELIM)
- **TOLERANCE LIMIT** (FELIM)

Trimming action:
- **SYNCERR**
- **SYNCH**
- **SYNCP**
- **SYNCP**

CRS event:
- **SYNCERR**
- **SYNCWARN**
- **SYNCP**
- **SYNCWARN**

Frequency error counter stopped

Frequency error counter stopped

ESYNC

Down

Up

+2 +1 0 -1 -2 0
9.5.3 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:
- TOLERANCE LIMIT, given directly in the FELIM field of the CRS_CFGCR register
- WARNING LIMIT, defined as 3 * FELIM value
- OUTRANGE (error limit), defined as 128 * FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.
  - SYNCOK status indicated
  - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCWARN status indicated
  - TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, it means that a stronger trimming action is necessary, and there is a risk that the optimal TRIM value will not be reached for the next period.
  - SYNCERR or SYNCMISS status indicated
  - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, it means that the frequency is out of the trimming range. This can also happen when the SYNC input is not clean or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  - SYNCERR or SYNCMISS status indicated
  - TRIM value not changed in AUTOTRIM mode

Note: If the actual value of the TRIM field is so close to its limits that the automatic trimming would force it to overflow or underflow, then the TRIM value is set just to the limit and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register), the TRIM field of CRS_CR is adjusted by hardware and is read-only.

9.5.4 CRS initialization and configuration

RELOAD value

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. It is then decreased by one to reach the expected synchronization on the zero value. The formula is the following:

\[
\text{RELOAD} = \left( \frac{f_{\text{TARGET}}}{f_{\text{SYNC}}} \right) - 1
\]

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).
FELIM value
The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of HSI48 oscillator clock ticks. The following formula can be used:

\[
\text{FELIM} = \left( \frac{f_{\text{TARGET}}}{f_{\text{SYNC}}} \right) \times \text{STEP} / 100 \times 2
\]

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to \( \left( \frac{f_{\text{TARGET}}}{f_{\text{SYNC}}} \right) = 48000 \) and to a typical trimming step size of 0.14%.

Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields which can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than 128 * FELIM value (OUTRANGE limit).

9.6 CRS low-power modes

Table 66. Effect of low-power modes on CRS

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. CRS interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI48 oscillator restarted.</td>
</tr>
<tr>
<td>Standby</td>
<td>The CRS peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

9.7 CRS interrupts

Table 67. Interrupt control bits

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Clear flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected synchronization</td>
<td>ESYNCF</td>
<td>ESYNCIE</td>
<td>ESYNCC</td>
</tr>
<tr>
<td>Synchronization OK</td>
<td>SYNCOKF</td>
<td>SYNCOKIE</td>
<td>SYNCOKC</td>
</tr>
<tr>
<td>Synchronization warning</td>
<td>SYNCWARNF</td>
<td>SYNCWARNIE</td>
<td>SYNCWARNC</td>
</tr>
<tr>
<td>Synchronization or trimming error</td>
<td>ERRF</td>
<td>ERRIE</td>
<td>ERRC</td>
</tr>
</tbody>
</table>
9.8 CRS registers

Refer to *Section 1.2 on page 97* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

### 9.8.1 CRS control register (CRS_CR)

Address offset: 0x00

Reset value: 0x0000 X000 (X=4 for products supporting 7-bit TRIM width, otherwise X=2)

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rt_w1</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bits 31:14** Reserved, must be kept at reset value.

**Bits 13:8** TRIM[5:0]: HSI48 oscillator smooth trimming

These bits provide a user-programmable trimming value to the HSI48 oscillator. They can be programmed to adjust to variations in voltage and temperature that influence the frequency of the HSI48 oscillator.

The default value is 32, which corresponds to the middle of the trimming interval. The trimming step is specified in the product datasheet. A higher TRIM value corresponds to a higher output frequency.

When the AUTOTRIMEN bit is set, this field is controlled by hardware and is read-only.

**Bit 7** SWSYNC: Generate software SYNC event

This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.

0: No action

1: A software SYNC event is generated.

**Bit 6** AUTOTRIMEN: Automatic trimming enable

This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to *Section 9.5.3* for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.

1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

**Bit 5** CEN: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled

1: Frequency error counter enabled

When this bit is set, the CRS_CFGR register is write-protected and cannot be modified.

**Bit 4** Reserved, must be kept at reset value.

**Bit 3** ESYNCIE: Expected SYNC interrupt enable

0: Expected SYNC (ESYNCF) interrupt disabled

1: Expected SYNC (ESYNCF) interrupt enabled
Bit 2 **ERRIE**: Synchronization or trimming error interrupt enable
0: Synchronization or trimming error (ERRF) interrupt disabled
1: Synchronization or trimming error (ERRF) interrupt enabled

Bit 1 **SYNCWARNIE**: SYNC warning interrupt enable
0: SYNC warning (SYNCWARNF) interrupt disabled
1: SYNC warning (SYNCWARNF) interrupt enabled

Bit 0 **SYNCOKIE**: SYNC event OK interrupt enable
0: SYNC event OK (SYNCOKF) interrupt disabled
1: SYNC event OK (SYNCOKF) interrupt enabled

### 9.8.2 CRS configuration register (CRS_CFGR)
This register can be written only when the frequency error counter is disabled (CEN bit is cleared in CRS_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

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

Bit 31 **SYNCPOL**: SYNC polarity selection
This bit is set and cleared by software to select the input polarity for the SYNC signal source.
0: SYNC active on rising edge (default)
1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **SYNC[SRC][1:0]**: SYNC signal source selection
These bits are set and cleared by software to select the SYNC signal source.
00: CRS_SYNC pin selected as SYNC signal source
01: LSE selected as SYNC signal source
10: OTG HS SOF selected as SYNC signal source (default)
11: Reserved

*Note:* When using USB LPM (Link Power Management) and the device is in Sleep mode, the periodic USB SOF will not be generated by the host. No SYNC signal will therefore be provided to the CRS to calibrate the HSI48 oscillator on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE clock or the SYNC pin must be used as SYNC signal.

Bit 27 Reserved, must be kept at reset value.
9.8.3 CRS interrupt and status register (CRS_ISR)

Address offset: 0x08
Reset value: 0x0000 0000

Bits 26:24 **SYNCDIV[2:0]**: SYNC divider

These bits are set and cleared by software to control the division factor of the SYNC signal.
- 000: SYNC not divided (default)
- 001: SYNC divided by 2
- 010: SYNC divided by 4
- 011: SYNC divided by 8
- 100: SYNC divided by 16
- 101: SYNC divided by 32
- 110: SYNC divided by 64
- 111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]**: Frequency error limit

FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS_ISR register. Refer to Section 9.5.3 for more details about FELIM evaluation.

Bits 15:0 **RELOAD[15:0]**: Counter reload value

RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to Section 9.5.2 for more details about counter behavior.

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

Bits 31:16 **FECAP[15:0]**: Frequency error capture

FECAP is the frequency error counter value latched in the time of the last SYNC event. Refer to Section 9.5.3 for more details about FECAP usage.

Bit 15 **FEDIR**: Frequency error direction

FEDIR is the counting direction of the frequency error counter latched in the time of the last SYNC event. It shows whether the actual frequency is below or above the target.
- 0: Upcounting direction, the actual frequency is above the target.
- 1: Downcounting direction, the actual frequency is below the target.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **TRIMOVF**: Trimming overflow or underflow

This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.
- 0: No trimming error signalized
- 1: Trimming error signalized
Bit 9  **SYNCMISS**: SYNC missed
This flag is set by hardware when the frequency error counter reached value FELIM * 128 and no SYNC was detected, meaning either that a SYNC pulse was missed or that the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, and that some other action has to be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC) and an interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.
0: No SYNC missed error signalized
1: SYNC missed error signalized

Bit 8  **SYNCERR**: SYNC error
This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.
0: No SYNC error signalized
1: SYNC error signalized

Bits 7:4  Reserved, must be kept at reset value.

Bit 3  **ESYNCF**: Expected SYNC flag
This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS_CR register. It is cleared by software by setting the ESYNCC bit in the CRS_ICR register.
0: No expected SYNC signalized
1: Expected SYNC signalized

Bit 2  **ERRF**: Error flag
This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS_ICR register.
0: No synchronization or trimming error signalized
1: Synchronization or trimming error signalized

Bit 1  **SYNCWARNF**: SYNC warning flag
This flag is set by hardware when the measured frequency error is greater than or equal to FELIM * 3, but smaller than FELIM * 128. This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS_ICR register.
0: No SYNC warning signalized
1: SYNC warning signalized

Bit 0  **SYNCOKF**: SYNC event OK flag
This flag is set by hardware when the measured frequency error is smaller than FELIM * 3. This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCOKKC bit in the CRS_ICR register.
0: No SYNC event OK signalized
1: SYNC event OK signalized
9.8.4 CRS interrupt flag clear register (CRS_ICR)

Address offset: 0x0C
Reset value: 0x0000 0000

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

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

Bit 3 **ESYNCC**: Expected SYNC clear flag
Writing 1 to this bit clears the ESYNCF flag in the CRS_ISR register.

Bit 2 **ERRC**: Error clear flag
Writing 1 to this bit clears TRIMOVF, SYNCMISS and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.

Bit 1 **SYNCWARNC**: SYNC warning clear flag
Writing 1 to this bit clears the SYNCWARNF flag in the CRS_ISR register.

Bit 0 **SYNCOKC**: SYNC event OK clear flag
Writing 1 to this bit clears the SYNCKOF flag in the CRS_ISR register.
## 9.8.5 CRS register map

### Table 68. CRS register map and reset values

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | CRS_CR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | CRS_CFGR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|        |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | CRS_ISR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | CRS_ICR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | CRS_CR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |          | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Refer to Section 2.3 on page 128 for the register boundary addresses.
10 Hardware semaphore (HSEM)

10.1 Hardware semaphore introduction

The hardware semaphore block provides 16 (32-bit) register based semaphores. The semaphores can be used to ensure synchronization between different processes running on the core. The HSEM provides a non blocking mechanism to lock semaphores in an atomic way. The following functions are provided:

- Locking a semaphore can be done in two ways:
  - 2-step lock: by writing MASTERID and PROCID to the semaphore, followed by a read check
  - 1-step lock: by reading the MASTERID from the semaphore
- Interrupt generation when a semaphore is freed
  - Each semaphore may generate an interrupt
- Semaphore clear protection
  - A semaphore is only cleared when MASTERID and PROCID match
- Global semaphore clear per MASTERID

10.2 Hardware semaphore main features

The HSEM includes the following features:

- 16 (32-bit) semaphores
- 8-bit PROCID
- 4-bit MASTERID
- 1 interrupt line
- Lock indication
10.3 HSEM functional description

10.3.1 HSEM block diagram

As shown in Figure 68, the HSEM is based on three sub-blocks:

- The semaphore block containing the semaphore status and IDs
- The semaphore interface block providing AHB access to the semaphore via the HSEM_Rx and HSEM_RLRx registers
- The interrupt interface block providing control for the interrupts via the HSEM_ISR, HSEM_IER, HSEM_MISR, and HSEM_ICR registers

Figure 68. HSEM block diagram

10.3.2 HSEM internal signals

Table 69. HSEM internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsem_hclk</td>
<td>Digital input</td>
<td>AHB clock</td>
</tr>
<tr>
<td>hsem_int_it</td>
<td>Digital output</td>
<td>Interrupt line</td>
</tr>
</tbody>
</table>
10.3.3 HSEM lock procedures

There are two lock procedures:

- 2-step (write) lock
- 1-step (read) lock

The semaphore is free when its LOCK bit is 0. In this case, the MASTERID and PROCID are also 0. When the LOCK bit is 1, the semaphore is locked and the MASTERID indicates which AHB bus master locked it. The PROCID indicates which process of that AHB bus master locked the semaphore.

When write locking a semaphore, the MASTERID is taken from the bus master ID and the PROCID is taken from the write data. When read locking the semaphore, the MASTERID is taken from the AHB bus master ID, and the PROCID is zero. There is no PROCID available with the 1-step (read) lock.

The MASTERID is taken from the AHB bus master ID. The PROCID is written by the software of that AHB bus master. Each AHB bus master process must have a unique PROCID. PROCID is only available in the 2-step lock procedure.

The two procedures (1-step and 2-step) can be used concurrently.

2-step (write) lock procedure

The 2-step lock procedure consists in a write to lock the semaphore, followed by a read to check if the lock has been successful, carried out from the HSEM_Rx register

- Write semaphore with PROCID and MASTERID, and LOCK = 1
  Lock is put in place when semaphore is free at write time.
- Read-back the semaphore
  The software checks the lock status, if PROCID and MASTERID match, then the lock is confirmed.
- Else retry (the semaphore has been locked by another AHB bus master or process)
  A semaphore can only be locked when it is free.
  A semaphore can be locked when the PROCID = 0.
Consecutive write attempts with LOCK = 1 to a locked semaphore, are ignored.
1-step (read) lock procedure

The 1-step procedure consists in a read to lock and check the semaphore in a single step from the HSEM_RLRx register.

- Read Lock semaphore with MASTERID.
- If read MASTERID matches and PROCID = 0, then the lock is put in place. If MASTERID matches and PROCID is not 0, this means that another process from the same MASTERID locked the semaphore with a 2-step (write) procedure.
- Else retry (the semaphore has been locked by another AHB bus master or process)

A semaphore can only be locked when it is free. When read locking a free semaphore, PROCID is 0. Read locking a locked semaphore returns the MASTERID and PROCID that locked it. All read locks, including the first one that locks the semaphore, return the MASTERID that locks or locked the semaphore.

If multiple processes of the same AHB bus master use the 1-step procedure, all processes using the same semaphore read the same status. When only one process locks the semaphore, each process of that AHB bus master reads the semaphore as locked by itself with the MASTERID.

10.3.4 HSEM write/read/readlock register address

For each semaphore, two AHB register addresses are provided, separated in two banks of 0x80.

In the first register address bank the semaphore can be written (locked/cleared) and read through the HSEM_Rx registers.

In the second register address bank the semaphore can be read (locked) through the HSEM_RLRx registers.

10.3.5 HSEM clear procedures

Clearing a semaphore is a protected process, to prevent accidental clearing by a AHB bus master or by a process that does not have the semaphore lock right. The semaphore clear procedure consists in writing to the semaphore with the corresponding MASTERID and PROCID and LOCK = 0. When cleared, the semaphore LOCK, the MASTERID, and the PROCID are all 0.

When cleared, an interrupt may be generated to signal the event. To this end, the semaphore interrupt must be enabled.

The clear procedure consists in a write to the semaphore HSEM_Rx registers:

- Write semaphore with PROCID and MASTERID, LOCK = 0.
- If PROCID and MASTERID match, semaphore is freed, and an interrupt may be generated when enabled.
- Else write is ignored, semaphore remains locked and no interrupt is generated (the semaphore is locked by another AHB bus master or process).

If multiple processes of the same AHB bus master use the 1-step lock procedure (PROCID = 0), all processes using the same semaphore clear the semaphore also for the other processes of that AHB bus master.
10.3.6 HSEM MASTERID semaphore clear

All semaphores locked by a AHB bus master can be cleared all at once by using the HSEM_CR register.

The procedure to clear all semaphores locked by a AHB bus master is the following:
- Write MASTERID and correct KEY value. All locked semaphore with a matching MASTERID are cleared (set to free), and may generate an interrupt when enabled.

This procedure may be used in case of an incorrect functioning AHB bus master, where the AHB bus master can free the locked semaphores by writing the MASTERID into the HSEM_CR register with the correct KEY value. This clears all locked semaphores with a matching MASTERID.

An interrupt may be generated for the semaphore(s) that become free. To this end, the semaphore interrupt must be enabled in the HSEM_IER registers.

10.3.7 HSEM interrupts

The hsem_int_it interrupt line allows each of the semaphores to generate an interrupt.

This interrupt line provides the following features:
- Interrupt enable per semaphore
- Interrupt clear per semaphore
- Interrupt status per semaphore
- Masked interrupt status per semaphore

With the interrupt enable (HSEM_IER), the semaphores affecting the interrupt line can be enabled. Disabled (masked) semaphore interrupts do not set the masked interrupt status for that semaphore, and do not generate an interrupt on the interrupt line.

The interrupt clear (HSEM_ICR) clears the interrupt status and masked interrupt status of the associated semaphore for the interrupt line.

The interrupt status (HSEM_ISR) mirrors the semaphore interrupt status of the interrupt line before the enable.

The masked interrupt status (HSEM_MISR) only mirrors the semaphore interrupt status of the enabled semaphore interrupts on the interrupt line. All masked interrupt status of the enabled semaphore must be cleared in order to clear the interrupt line.
The procedure to get an interrupt when a semaphore becomes free is described hereafter.

**Try to lock the semaphore x**
- If the semaphore lock is obtained, no interrupt is needed.
- If the semaphore lock fails:
  - Clear pending semaphore x interrupt status for the interrupt line hsem_int_it in HSEM_ICR.
  - Re-try to lock the semaphore x again:
    - If the semaphore lock is obtained, no interrupt is needed (semaphore has been freed between first try to lock it and clear semaphore interrupt status).
    - If the semaphore lock fails, enable the semaphore x interrupt for the interrupt line hsem_int_it in HSEM_IER.

![Figure 70. Interrupt state diagram](image-url)
On the semaphore x free interrupt, try to lock the semaphore x

- If the semaphore lock is obtained:
  Disable the semaphore x interrupt for the interrupt line hsem_int_it in HSEM_IER.
  Clear pending semaphore x interrupt status for the interrupt line hsem_int_it in HSEM_ICR.

- If the semaphore x lock fails:
  Clear pending semaphore x interrupt status for the interrupt line hsem_int_it in HSEM_ICR.
  Try again to lock the semaphore x:
    - If the semaphore lock is obtained (semaphore has been freed between first try to lock and semaphore Interrupt status clear), Disable the semaphore interrupt for the interrupt line hsem_int_it in HSEM_IER.
    - If the semaphore lock failed, wait for semaphore free interrupt.

Note: An interrupt does not lock the semaphore. After an interrupt, either the AHB bus master or the process must still perform the lock procedure to lock the semaphore.

10.3.8 AHB bus master ID verification

The HSEM allows only authorized AHB bus master IDs to lock and unlock semaphores.

- The AHB bus master 2-step lock write access to the semaphore HSEM_Rx register is checked against the valid bus master IDs.
  - Accesses from unauthorized AHB bus master IDs are discarded and do not lock the semaphore.

- The AHB bus master 1-step lock read access from the semaphore HSEM_RLRx register is checked against the valid bus master IDs.
  - An unauthorized AHB bus master ID read from HSEM_RLRx returns the following Read data depending on the semaphore status:
    - when the semaphore is free, it returns all 0.
    - when the semaphore has been locked before, it returns the HSEM_RLRx data.

- The MASTERID semaphore clear write access to the HSEM_CR register is checked against the valid bus master IDs. Only the valid bus master IDs can write to the HSEM_CR register and clear any of the MASTERID semaphores.
  - Accesses from unauthorized AHB bus master IDs are discarded and do not clear the MASTERID semaphore.

Table 70 details the relation between bus master/CPU and MASTERID.

<table>
<thead>
<tr>
<th>Table 70. Authorized AHB bus master ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus master 0 (CPU)</td>
</tr>
<tr>
<td>MASTERID = 1</td>
</tr>
</tbody>
</table>

Note: Accesses from unauthorized AHB bus master IDs to other registers are granted.
10.4 **HSEM registers**

Registers must be accessed using word format. Byte and half-word accesses are ignored and have no effect on the semaphores. Byte and half-word read accesses always return 0. Byte and half-word accesses do not generate a bus error.

10.4.1 **HSEM register semaphore x (HSEM_Rx)**

Address offset: 0x000 + 0x4 * x (x = 0 to 15)

Reset value: 0x0000 0000

The HSEM_Rx must be used to perform a 2-step write lock and read back. Only write accesses with authorized AHB bus master IDs are granted. Write accesses with unauthorized AHB bus master IDs are discarded.

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

Bit 31 **LOCK**: Lock indication

This bit can be written and read by software.

0: On write free semaphore (only when MASTERID and PROCID match), on read semaphore is free.

1: On write try to lock semaphore, on read semaphore is locked.

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 **MASTERID[3:0]**: Semaphore MASTERID

Written by software, when the semaphore is free and the LOCK bit is at the same time written to 1, the MASTERID is written only when the bus ID of the AHB bus master writing the semaphore matches.

When the semaphore is cleared (LOCK bit written to 0 and AHB bus master ID matched MASTERID), the MASTERID is cleared to 0.

When the semaphore is cleared (LOCK bit written to 0 and AHB bus master ID does not match MASTERID), the MASTERID is not be affected.

Write when LOCK bit is already 1 (semaphore locked), the MASTERID is not affected.

A read returns the stored MASTERID value.

Bits 7:0 **PROCID[7:0]**: Semaphore PROCID

Written by software, when the semaphore is free and the lock bit is written to 1, the PROCID is set to the written data.

When the semaphore is cleared, (LOCK bit written to 0), the PROCID is cleared to 0.

Write when LOCK bit is already 1 (semaphore locked), the PROCID is not affected.

A read returns the programmed PROCID value.
10.4.2 HSEM read lock register semaphore x (HSEM_RLRx)

Address offset: 0x080 + 0x004 * x (x = 0 to 15)

Reset value: 0x0000 0000

Accesses the same physical bits as HSEM_Rx. The HSEM_RLRx must be used to perform a 1-step read lock. Only read accesses with authorized AHB bus master IDs are granted. Read accesses with unauthorized AHB bus master IDs are discarded.

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

Bit 31 **LOCK**: Lock indication
This bit is read only by software at this address. A read with a valid bus master ID always returns 1.
When the semaphore is free and the software performs a read, the hardware sets the semaphore to locked.
When the semaphore is locked and the software performs a read, the LOCK bit is not effected.
0: Semaphore free
1: Semaphore locked

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 **MASTERID[3:0]**: Semaphore MASTERID
This field is read only by software at this address.
On a read, when the semaphore is free, the hardware sets the MASTERID to the AHB bus master ID reading the semaphore. The MASTERID of the AHB bus master locking the semaphore, is read.
On a read when the semaphore is locked, this field returns the MASTERID of the AHB bus master that has locked the semaphore.

Bits 7:0 **PROCID[7:0]**: Semaphore processor ID
This field is read only by software at this address.
On a read when the semaphore is free, this field returns 0.
On a read when the semaphore is locked, this field returns the processor ID of the process that locked the semaphore.
## 10.4.3 HSEM interrupt enable register (HSEM_IER)

Address offset: 0x100  
Reset value: 0x0000 0000

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ISE[15:0]**: Interrupt semaphore x enable bit  
This bit is read and written by software.  
0: Interrupt generation for semaphore x disabled (masked)  
1: Interrupt generation for semaphore x enabled (not masked)

## 10.4.4 HSEM interrupt clear register (HSEM_ICR)

Address offset: 0x104  
Reset value: 0x0000 0000

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ISC[15:0]**: Interrupt semaphore x clear bit  
This bit is read and written by software, and is always read 0.  
0: Interrupt semaphore x status and masked status not affected  
1: Interrupt semaphore x status and masked status cleared

## 10.4.5 HSEM interrupt status register (HSEM_ISR)

Address offset: 0x108  
Reset value: 0x0000 0000

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

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

**ISR[15:0]**

498/2974  
RM0455 Rev 3
### 10.4.6 HSEM interrupt status register (HSEM_MISR)

Address offset: 0x10C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>ISF[15:0]: Interrupt semaphore x status bit before enable (mask)</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

### 10.4.7 HSEM clear register (HSEM_CR)

Address offset: 0x140

Reset value: 0x0000 0000

Only write accesses with authorized AHB bus master IDs are granted. Write accesses with unauthorized AHB bus master IDs are discarded.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 MISF[15:0]: Masked interrupt semaphore x status bit after enable (mask)

0: Interrupt semaphore x status after masking not pending
1: Interrupt semaphore x status after masking pending
Bits 31:16 **KEY[15:0]:** Semaphore clear key

This field can be written by software and is always read 0.
If this key value does not match HSEM_KEYR.KEY, semaphores are not effected.
If this key value matches HSEM_KEYR.KEY, all semaphores matching the MASTERID are cleared to the free state.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **MASTERID[3:0]:** MASTERID of semaphores to be cleared

This field can be written by software and is always read 0.
This field indicates the MASTERID for which the semaphores are cleared when writing the HSEM_CR.

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

### 10.4.8 HSEM interrupt clear register (HSEM_KEYR)

Address offset: 0x144

Reset value: 0x0000 0000

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

Bits 31:16 **KEY[15:0]:** Semaphore clear key

This field can be written and read by software.
Key value to match when clearing semaphores.

Bits 15:0 Reserved, must be kept at reset value.
### 10.4.9 HSEM register map

Table 71. HSEM register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | HSEM_R0       | LOCK | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Master value  | 0    | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x004  | HSEM_R1       | LOCK | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Master value  | 0    | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x03C  | HSEM_R15      | LOCK | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Master value  | 0    | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x080  | HSEM_RLR0     | LOCK | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Master value  | 0    | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x084  | HSEM_RLR1     | LOCK | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Master value  | 0    | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0BC  | HSEM_RLR15    | LOCK | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Master value  | 0    | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x100  | HSEM_IER      | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Master value  | 00000000000000000000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x104  | HSEM_ICR      | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Master value  | 00000000000000000000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x108  | HSEM_ISR      | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Master value  | 00000000000000000000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x10C  | HSEM_MISR     | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Master value  | 00000000000000000000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x140  | HSEM_CR       | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Master value  | 00000000000000000000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x144  | HSEM_KEYR     | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Master value  | 00000000000000000000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Refer to Section 2.3 on page 128 for the register boundary addresses.
11 General-purpose I/Os (GPIO)

11.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR) and a 32-bit set/reset register (GPIOx_BSRR). In addition all GPIOs have a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

11.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

11.3 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIOx_BSRR register is to allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.
**Figure 71** and **Figure 72** show the basic structures of a standard and a 5-Volt tolerant I/O port bit, respectively. **Table 72** gives the possible port bit configurations.

**Figure 71. Basic structure of an I/O port bit**

**Figure 72. Basic structure of a 5-Volt tolerant I/O port bit**

1. \( V_{DD\_FT} \) is a potential specific to five-volt tolerant I/Os and different from \( V_{DD} \).
### Table 72. Port bit configuration table(1)

<table>
<thead>
<tr>
<th>MODE(i) [1:0]</th>
<th>OTYPE(i)</th>
<th>OSPEED(i) [1:0]</th>
<th>PUPD(i) [1:0]</th>
<th>I/O configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>GP output PP</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>GP output PP + PU</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>GP output PP + PD</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>GP output OD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>GP output OD + PU</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>GP output OD + PD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Reserved (GP output OD)</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>AF output PP</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>AF output PP + PU</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>AF output PP + PD</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>AF output OD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>AF output OD + PU</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>AF output OD + PD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>Input Floating</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>Input PU</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>Input PD</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>Reserved (input floating)</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>Input/output Analog</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.
11.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in input floating mode.

The debug pins are in AF pull-up/pull-down after reset:
- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDAT in pull-up
- PB4: NJTRST in pull-up
- PB3: JTD0 in floating state

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIOx_PUPDR register.

11.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to sixteen alternate function inputs (AF0 to AF15) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:
- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.
- Cortex-M7 with FPU EVENTOUT is mapped on AF15

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user has to proceed as follows:
- **Debug function**: after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host
- **System function**: MCOx pins have to be configured in alternate function mode.
- **GPIO**: configure the desired I/O as output, input or analog in the GPIOx_MODER register.

- **Peripheral alternate function**:
  - Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDER registers, respectively.
– Configure the desired I/O as an alternate function in the GPIOx_MODER register.

**Additional functions:**
– For the ADC, DAC and OPAMP, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC, DAC and OPAMP registers.

As indicated above, for the additional functions (such as DAC or OPAMP), the output is controlled by the corresponding peripheral. Care must be taken to select the I/O port analog function before enabling the additional function output in the peripheral control register.

– For the additional functions like RTC_OUT, RTC_TS, RTC_TAMPx, WKUPx and oscillators, configure the required function in the related RTC, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers. For details about I/O control by the RTC, refer to Section 50.3: RTC functional description on page 1818.

**EVENTOUT**
– Configure the I/O pin used to output the core EVENTOUT signal by connecting it to AF15.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

### 11.3.3 I/O port control registers
Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

### 11.3.4 I/O port data registers
Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR). GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.

See Section 11.4.5: GPIO port input data register (GPIOx_IDR) (x = A to K) and Section 11.4.6: GPIO port output data register (GPIOx_ODR) (x = A to K) for the register descriptions.

### 11.3.5 I/O data bitwise handling
The bit set reset register (GPIOx_BSRR) is a 32-bit register which allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, bit BS(i) sets the corresponding ODR(i) bit. When written to 1, bit BR(i) resets the ODR(i) corresponding bit.
Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: it is possible to modify one or more bits in a single atomic AHB write access.

### 11.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write / read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence has been applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence (refer to Section 11.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to K)) can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details refer to LCKR register description in Section 11.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to K).

### 11.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin refer to the device datasheet.

### 11.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode.

Refer to Section 20: Extended interrupt and event controller (EXTI).
11.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register provides the I/O state

*Figure 73* shows the input configuration of the I/O port bit.

**Figure 73. Input floating/pull up/pull down configurations**

11.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state
- A read access to the output data register gets the last written value

*Figure 74* shows the output configuration of the I/O port bit.
11.3.11 I/O compensation cell

This cell is used to control the I/O commutation slew rate ($t_{\text{fall}} / t_{\text{rise}}$) to reduce the I/O noise on power supply.

The cell is split into two blocks:

- The first block provides an optimal code for the current PVT. The code stored in this block can be read when the READY flag of the SYSCFG_CCSR is set.
- The second block controls the I/O slew rate. The user selects the code to be applied and programs it by software.

The I/O compensation cell features 2 voltage ranges: 1.62 to 2.0 V and 2.7 to 3.6 V.

11.3.12 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data)
- The Schmitt trigger input is activated
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state

*Figure 75* shows the Alternate function configuration of the I/O port bit.
11.3.13 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up and pull-down resistors are disabled by hardware
- Read access to the input data register gets the value “0”

*Figure 76* shows the high-impedance, analog-input configuration of the I/O port bits.

---

Some pins/balls are directly connected to PA0_C, PA1_C, PC2_C and PC3_C ADC analog inputs (see *Figure 77*): there is a direct path between Pxy_C and Pxy pins/balls, through an...
analog switch (refer to Section 12.4.1: SYSCFG peripheral mode configuration register (SYSCFG_PMCR) for details on how to configure analog switches).

**Figure 77. Analog inputs connected to ADC inputs**

1. VDD_FT is a potential specific to 5V tolerant I/Os. It is distinct from VDD.

### 11.3.14 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched OFF (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched ON (by setting the HSEON or LSEON bit in the RCC_CSR register) the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the OSC_IN or OSC32_IN pin is reserved for clock input and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

### 11.3.15 Using the GPIO pins in the backup supply domain

The PC13/PC14/PC15/PI8 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.
11.4  GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to Table 73.

The peripheral registers can be written in word, half word or byte mode.

11.4.1  GPIO port mode register (GPIOx_MODER) (x =A to K)

Address offset:0x00

Reset value: 0xABFF FFFF for port A

Reset value: 0xFFFF FEBF for port B

Reset value: 0xFFFF FFFF for other ports

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

Bits 31:0  MODER[15:0][1:0]: Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

11.4.2  GPIO port output type register (GPIOx_OTYPER) (x = A to K)

Address offset: 0x04

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>OT15</th>
<th>OT14</th>
<th>OT13</th>
<th>OT12</th>
<th>OT11</th>
<th>OT10</th>
<th>OT9</th>
<th>OT8</th>
<th>OT7</th>
<th>OT6</th>
<th>OT5</th>
<th>OT4</th>
<th>OT3</th>
<th>OT2</th>
<th>OT1</th>
<th>OT0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
11.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to K)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)
Reset value: 0x0000 00C0 (for port B)
Reset value: 0x0000 0000 (for other ports)

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

Bits 31:0 OSPEEDR[15:0]: Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O output speed.

00: Low speed
01: Medium speed
10: High speed
11: Very high speed

Note: Refer to the product datasheets for the values of OSPEEDRy bits versus VDD range and external load.

11.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to K)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)
Reset value: 0x0000 0100 (for port B)
Reset value: 0x0000 0000 (for other ports)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
11.4.5 GPIO port input data register (GPIOx_IDR)
(x = A to K)

Address offset: 0x10
Reset value: 0x0000 XXXX

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

Bits 31:16 Reserved, must be kept at reset value.
Bits 15:0 IDR[15:0]: Port x input data I/O pin y (y = 15 to 0)
These bits are read-only. They contain the input value of the corresponding I/O port.

11.4.6 GPIO port output data register (GPIOx_ODR)
(x = A to K)

Address offset: 0x14
Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.
Bits 15:0 ODR[15:0]: Port output data I/O pin y (y = 15 to 0)
These bits can be read and written by software.
Note: For atomic bit set/reset, the ODR bits can be individually set and/or reset by writing to the GPIOx_BSRR register (x = A..F).

11.4.7 GPIO port bit set/reset register (GPIOx_BSRR)
(x = A to K)

Address offset: 0x18
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR15</td>
<td>BR14</td>
<td>BR13</td>
<td>BR12</td>
<td>BR11</td>
<td>BR10</td>
<td>BR9</td>
<td>BR8</td>
<td>BR7</td>
<td>BR6</td>
<td>BR5</td>
<td>BR4</td>
<td>BR3</td>
<td>BR2</td>
<td>BR1</td>
<td>BR0</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>BS15</td>
<td>BS14</td>
<td>BS13</td>
<td>BS12</td>
<td>BS11</td>
<td>BS10</td>
<td>BS9</td>
<td>BS8</td>
<td>BS7</td>
<td>BS6</td>
<td>BS5</td>
<td>BS4</td>
<td>BS3</td>
<td>BS2</td>
<td>BS1</td>
<td>BS0</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

 Bits 31:16 **BR[15:0]**: Port x reset I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODRx bit
1: Resets the corresponding ODRx bit
**Note:** If both BSx and BRx are set, BSx has priority.

 Bits 15:0 **BS[15:0]**: Port x set I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODRx bit
1: Sets the corresponding ODRx bit

11.4.8 **GPIO port configuration lock register (GPIOx_LCKR)**
(x = A to K)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

**Note:** A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCK15</td>
<td>LCK14</td>
<td>LCK13</td>
<td>LCK12</td>
<td>LCK11</td>
<td>LCK10</td>
<td>LCK9</td>
<td>LCK8</td>
<td>LCK7</td>
<td>LCK6</td>
<td>LCK5</td>
<td>LCK4</td>
<td>LCK3</td>
<td>LCK2</td>
<td>LCK1</td>
<td>LCK0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Bits 31:17  Reserved, must be kept at reset value.

Bit 16  **LCKK**: Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active
1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

**LOCK key write sequence:**

WR LCKR[16] = ‘1’ + LCK[15:0]
WR LCKR[16] = ‘0’ + LCKR[15:0]
WR LCKR[16] = ‘1’ + LCKR[15:0]
RD LCKR

RD LCKR[16] = ‘1’ (this read operation is optional but it confirms that the lock is active)

*Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.*

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit returns ‘1’ until the next MCU reset or peripheral reset.

Bits 15:0  **LCK[15:0]**: Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is ‘0’.

0: Port configuration not locked
1: Port configuration locked

### 11.4.9  GPIO alternate function low register (GPIOx_AFRL)  
(x = A to K)

Address offset: 0x20

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
11.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to J)

Address offset: 0x24
Reset value: 0x0000 0000

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

Bits 31:0 **AFR[7:0][3:0]**: Alternate function selection for port x I/O pin y (y = 7 to 0)

These bits are written by software to configure alternate function I/Os.

0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15
Bits 31:0  \textbf{AFR[15:8][3:0]}: Alternate function selection for port \(x\) I/O pin \(y\) (\(y = 15\) to 8)

These bits are written by software to configure alternate function I/Os.

- 0000: AF0
- 0001: AF1
- 0010: AF2
- 0011: AF3
- 0100: AF4
- 0101: AF5
- 0110: AF6
- 0111: AF7
- 1000: AF8
- 1001: AF9
- 1010: AF10
- 1011: AF11
- 1100: AF12
- 1101: AF13
- 1110: AF14
- 1111: AF15
### 11.4.11 GPIO register map

The following table gives the GPIO register map and reset values.

#### Table 73. GPIO register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | GPIOA_MODER   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 1  | 0  | 1  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x00   | GPIOB_MODER   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 0  | 1  | 0  | 1  | 1  | 1  | 1  | 1  |
| 0x00   | GPIOx_MODER   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | (where x = C..K) |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x04   | GPIOx_OTYPER  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | (where x = A to K) |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x08   | GPIOA_OSPEEDR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x08   | GPIOB_OSPEEDR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x08   | GPIOx_OSPEEDR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | (where x = C..K) |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0C   | GPIOA_PUPDR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 1  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
Refer to Section 2.3 on page 128 for the register boundary addresses.
12 System configuration controller (SYSCFG)

12.1 Introduction

The devices feature a set of configuration registers. The objectives of this section is to describe in details the system configuration controller.

12.2 SYSCFG main features

The system configuration controller main functions are the following:

- Analog switch configuration management
- I2C Fm+ configuration
- Timer break input lockup management
- Management of the external interrupt line connection to the GPIOs
- Management of I/O speed in low-voltage mode
- Management of the I/O compensation cell for VDD and VDDMMC dual-power rail

12.3 SYSCFG functional description

12.3.1 Analog switch configuration management

To optimize the ADC performance, PA0_C, PA1_C, PC2_C and PC3_C pads are directly connected to the ADC (see Figure 77: Analog inputs connected to ADC inputs).

The switch connecting Pxx_C and Pxx pads is controlled through the SYSCFG_PMCR register (see Section 12.4.1).

Some packages feature Pxx_C pads but not Pxx. All Pxx alternate functions are available on Pxx_C pads by closing the switch.

Other packages feature Pxx pads but not Pxx_C pads. ADC applications can benefit from closing the switch and using an ADC fast channel input (see Figure 156: ADC1 connectivity and Figure 157: ADC2 connectivity).

12.3.2 I2C Fm+ configuration

The I2C Fast-mode Plus (see Section 52: Inter-integrated circuit (I2C) interface) requires to set the corresponding GPIOs in high driving mode.

The I2CxFMP bits of SYSCFG_PMCR register activate the Fast-mode Plus driving capability for the I2Cx GPIOs.

In addition, the high drive mode can be activated on some GPIOs independently from the I2C (through PxxFMP bits in SYSCFG_PMCR register). This can be useful to drive LEDs directly.
12.3.3 Timer break input lockup management

In addition to HSE break detection, this feature allows the disabling of the timer output when an ECC double error or a core lockup is detected. This is particularly useful when using timers to drive motors. In addition to HSE break detection, the lockup can be selected through the dedicated SYSCFG_CFGR register (see Section 12.4.6).

Note: The SYSCFG_CFGR register bits can be cleared only by a system reset.

Figure 78. Timer break input lockup block diagram

12.3.4 Management of external interrupt line connections to GPIOs

Any GPIO can be configured as an external interrupt by programming the SYSCFG_EXTICRx registers (refer also to Section 11.3.8: External interrupt/wakeup lines).

12.3.5 I/O speed in low-voltage mode

The speed of some I/Os can be increased at low voltage by programming the corresponding HSLVx bits in the SYSCFG_CCCSR register. This feature is intended mainly for SDMMC, FMC and OCTOSPI interfaces.

Caution: This feature must be used only when the I/O power supply is below 2.5 V.

12.3.6 I/O compensation cell management

The cell generates an 8-bit value for the I/O buffer (4-bit for N-MOS and 4-bit for P-MOS) that depends on PVT operating conditions (process, voltage, temperature). These bits are used to control the current slew-rate and output impedance in the I/O buffer.

By default, the compensation cell is disabled, and a fixed code is applied to all the I/Os.

When enabled, the compensation cell tracks the PVT, and the 8-bit code (PCV and NCV) is applied to all I/Os for both VDD and VDDMMC I/O supply domains.

To optimize trimming, the code can be adjusted through SYSCFG_CCCSR register. Two sets of bitfields are available: PCC/NCC for the VDD power rail and PCC_MMC/NCC_MCC for the VDDMMC power rail. They can be selected independently through CS and CS_MMC bits of SYSCFG_CCCSR register (see Figure 79).
When the two power rails are not at the same voltage, it is recommended to adjust VDDMMC code through SYSCFG_CCCR compensation cell code register since the compensation cell tracks the PVT on VDD power supply.

Figure 79. I/O compensation cell block diagram

12.4 SYSCFG registers

12.4.1 SYSCFG peripheral mode configuration register (SYSCFG_PMCR)

Address offset: 0x04
Reset value: 0x0F00 0000

<table>
<thead>
<tr>
<th>Bit 31-30</th>
<th>Bit 29-28</th>
<th>Bit 27-26</th>
<th>Bit 25-24</th>
<th>Bit 23-22</th>
<th>Bit 21-20</th>
<th>Bit 19-18</th>
<th>Bit 17-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Res</td>
<td>PC3SO</td>
<td>PC2SO</td>
<td>PA1SO</td>
<td>PA0SO</td>
<td>PC3SO</td>
<td>PC2SO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15  14  13 12  11  10  9  8  7  6  5  4  3  2  1  0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **PC3SO**: PC3 switch open
This bit controls the analog switch between PC3 and PC3_C (dual pad).
0: Analog switch closed (pads connected through the analog switch)
1: Analog switch open (two separated pads)

Bit 26 **PC2SO**: PC2 switch open
This bit controls the analog switch between PC2 and PC2_C (dual pad).
0: Analog switch closed (pads connected through the analog switch)
1: Analog switch open (two separated pads)

Bit 25 **PA1SO**: PA1 switch open
This bit controls the analog switch between PA1 and PA1_C (dual pad).
0: Analog switch closed (pads connected through the analog switch)
1: Analog switch open (two separated pads)

Bit 24 **PA0SO**: PA0 switch open
This bit controls the analog switch between PA0 and PA0_C (dual pad).
0: Analog switch closed (pads connected through the analog switch)
1: Analog switch open (two separated pads)

Bits 23:8 Reserved, must be kept at reset value.

Bit 7 **PB9FMP**: Fast-mode Plus driving capability activation on PB9
This bit enables the Fm+ driving mode on PB9.
0: PB9 pin operates in standard mode
1: Fm+ mode is enabled on PB9 pin and the speed control is bypassed

Bit 6 **PB8FMP**: Fast-mode Plus driving capability activation on PB8
This bit enables the Fm+ driving mode on PB8.
0: PB8 pin operates in standard mode
1: Fm+ mode is enabled on PB8 pin and the speed control is bypassed

Bit 5 **PB7FMP**: Fast-mode Plus driving capability activation on PB7
This bit enables the Fm+ driving mode for PB7.
0: PB7 pin operates in standard mode
1: Fm+ mode is enabled on PB7 pin and the speed control is bypassed

Bit 4 **PB6FMP**: Fast-mode Plus driving capability activation on PB6
This bit enables the Fm+ driving mode for PB6.
0: PB6 pin operates in standard mode
1: Fm+ mode is enabled on PB6 pin and the speed control is bypassed

Bit 3 **I2C4FMP**: I2C4 Fm+
This bit enables Fm+ on I2C4.
The corresponding GPIOs are configured in Fast-mode Plus driving capability.
0: Fm+ disabled
1: Fm+ enabled
Bit 2  **I2C3FMP**: I2C3 Fm+
   This bit enables Fm+ on I2C3.
   The corresponding GPIOs are configured in Fast-mode Plus driving capability
   0: Fm+ disabled
   1: Fm+ enabled

Bit 1  **I2C2FMP**: I2C2 Fm+
   This bit enables Fm+ on I2C2.
   The corresponding GPIOs are configured in Fast-mode Plus driving capability
   0: Fm+ disabled
   1: Fm+ enabled

Bit 0  **I2C1FMP**: I2C1 Fm+
   This bit enables Fm+ on I2C1.
   The corresponding GPIOs are configured in Fast-mode Plus driving capability
   0: Fm+ disabled
   1: Fm+ enabled

### 12.4.2 SYSCFG external interrupt configuration register 1
   **(SYSCFG_EXTICR1)**

Address offset: 0x08
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 0 to 3)
   These bits are written by software to select the source input for the EXTI input for external interrupt / event detection.
   0000: PA[x] pin
   0001: PB[x] pin
   0010: PC[x] pin
   0011: PD[x] pin
   0100: PE[x] pin
   0101: PF[x] pin
   0110: PG[x] pin
   0111: PH[x] pin
   1000: PI[x] pin
   1001: PJ[x] pin
   1010: PK[x] pin
   Other configurations: reserved
12.4.3  SYSCFG external interrupt configuration register 2
(SYSCFG_EXTICR2)

Address offset: 0x0C
Reset value: 0x0000 0000

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

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

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **EXTI[x][3:0]:** EXTI x configuration (x = 4 to 7)

These bits are written by software to select the source input for the EXTI input for
external interrupt / event detection.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin
1001: PJ[x] pin
1010: PK[x] pin

Other configurations: reserved
12.4.4  SYSCFG external interrupt configuration register 3
(SYSCFG_EXTICR3)

Address offset: 0x10
Reset value: 0x0000 0000

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

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


Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  EXTI[x][3:0]: EXTI x configuration (x = 8 to 11)

These bits are written by software to select the source input for the AEIC input for external interrupt / event detection.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin
1001: PJ[x] pin
1010: PK[x] pin

Other configurations: reserved

Note: PK[11:8] are not used.
12.4.5  SYSCFG external interrupt configuration register 4  
(SYSCFG_EXTICR4)

Address offset: 0x14
Reset value: 0x0000 0000

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

- Bits 31:16: Reserved, must be kept at reset value.
- Bits 15:0  **EXTIx[3:0]:** EXTI x configuration (x = 12 to 15)
  These bits are written by software to select the source input for the AEIC input for external interrupt / event detection.
  - 0000: PA[x] pin
  - 0001: PB[x] pin
  - 0010: PC[x] pin
  - 0011: PD[x] pin
  - 0100: PE[x] pin
  - 0101: PF[x] pin
  - 0110: PG[x] pin
  - 0111: PH[x] pin
  - 1001: PJ[x] pin
  - 1010: PK[x] pin
  - Other configurations: reserved

  *Note: PK[15:12] are not used.*

12.4.6  SYSCFG timer break lockup register  
(SYSCFG_CFGR)

Address offset: 0x18
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
</tr>
</tbody>
</table>
Bit 15  Reserved, must be kept at reset value.

Bit 14  **ITCML**: ITCM double ECC error signal lock
         This bit is set by software and cleared only by a reset system.
         It is used to enable and lock the ITCM double ECC error signal connection to TIM1/8/15/16/17 Break inputs.
         0: ITCM double ECC disconnected
         1: ITCM double ECC connected

Bit 13  **DTCML**: D1TCM or D0TCM double ECC error signal lock
         This bit is set by software and cleared only by reset system.
         It is used to enable and lock the DxTCM double ECC error signal connection to TIM1/8/15/16/17 Break inputs.
         0: DxTCM double ECC disconnected
         1: DxTCM double ECC connected

Bits 12:7  Reserved, must be kept at reset value.

Bit 6  **CM7L**: Cortex®-M7 LOCKUP (HardFault) output enable bit
         This bit is set by software and cleared only by reset system.
         It is used to enable and lock the Cortex®-M7 LOCKUP output connection to TIM1/8/15/16/17 Break inputs.
         0: Cortex®-M7 LOCKUP output disconnected
         1: Cortex®-M7 LOCKUP output connected

Bits 5:4  Reserved, must be kept at reset value.

Bit 3  **FLASHL**: Flash double ECC error lock bit
         This bit is set by software and cleared only by reset system.
         It is used to enable and lock the Cortex®-M7 LOCKUP output connection to TIM1/8/15/16/17 Break inputs.
         0: Double ECC disconnected
         1: Double ECC connected

Bit 2  **PVDL**: PVD lock enable bit.
         This bit is set by software and cleared only by reset system.
         It is used to enable and lock the PVD connection to TIM1/8/15/16/17 Break inputs.
         0: PVD signal disconnected
         1: PVD signal connected

Bits 1:0  Reserved, must be kept at reset value.
### 12.4.7 SYSCFG compensation cell control/status register (SYSCFG_CCCSR)

Address offset: 0x20  
Reset value: 0x0000 0000

Refer to Section 11.3.11: I/O compensation cell for a detailed description of I/O compensation mechanism.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Address Offset</th>
<th>Bit 31:17 Reserved, must be kept at reset value.</th>
<th>Bit 19 HSLV3: High-speed at low-voltage for VDDMMC I/Os</th>
<th>Bit 18 HSLV2: High-speed at low-voltage for VDD I/Os</th>
<th>Bit 17 HSLV1: High-speed at low-voltage for VDD I/Os</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>Bit 19 HSLV3: High-speed at low-voltage for VDDMMC I/Os</td>
<td>This bit is written by software to optimize the I/O speed when the product voltage is low. It is active only if VDDMMC_HSLV user option bit is set. It must be used only when the VDDMMC I/O power supply is below 2.5 V. Setting this bit when VDDMMC is higher than 2.5 V might damage the I/Os. It controls the speed of the following I/Os (mainly SDIO on VDDMMC power rail): PB[9:8], PD[7:6] and PG[14:9]. 0: No I/O speed optimization 1: I/O speed optimization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit 18 HSLV2: High-speed at low-voltage for VDD I/Os</td>
<td>This bit is written by software to optimize the I/O speed when the product voltage is low. It is active only if VDD_HSLV user option bit is set. It must be used only when the VDD I/O power supply is below 2.5 V. Setting this bit when VDD is higher than 2.5 V might damage the I/Os. This bit controls the speed of the following I/Os (mainly for FMC): PD[15:8], PD[3], PD[1:0], PE[15:0], PF[15:12], PF[5:2], PG[8:0], PH[15:8], PH[3:2], PI[10:9] and PI[7:0]. 0: No I/O speed optimization 1: I/O speed optimization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit 17 HSLV1: High-speed at low-voltage for VDD I/Os</td>
<td>This bit is written by software to optimize the I/O speed when the product voltage is low. It is active only if VDD_HSLV user option bit is set. It must be used only when the VDD I/O power supply is below 2.5 V. Setting this bit when VDD is higher than 2.5 V might damage the I/Os. It control the speed of the following I/Os (mainly for OCTOSPI): PA[7:6], PA[3], PA[1], PB[12], PB[2], PC[5], PD[5:4], PF[11:6], PG[15], PI[4:2], PJ[2:1] and PK[6:3]. 0: No I/O speed optimization 1: I/O speed optimization</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
12.4.8 SYSCFG compensation cell value register (SYSCFG_CCVR)

Address offset: 0x24
Reset value: 0x0000 0088
12.4.9 SYSCFG compensation cell code register (SYSCFG_CCCR)

Address offset: 0x28

Reset value: 0x0000 0000

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **PCC_MMC[3:0]**: PMOS compensation code register
These bits are written by software to define an I/O compensation cell code for PMOS transistors corresponding to the VDDMMC power rail.
This code is applied to the I/O when the CS_MMC bit of SYSCFG_CCCSR is set.

Bits 11:8 **NCC_MMC[3:0]**: NMOS compensation code register
These bits are written by software to define an I/O compensation cell code for NMOS transistors corresponding to the VDDMMC power rail.
This code is applied to the I/O when the CS_MMC bit of SYSCFG_CCCSR is set.

Bits 7:4 **PCC[3:0]**: PMOS compensation code register
These bits are written by software to define an I/O compensation cell code for PMOS transistors corresponding to the VDD power rail.
This code is applied to the I/O when the CS bit of SYSCFG_CCCSR is set.

Bits 3:0 **NCC[3:0]**: NMOS compensation code register
These bits are written by software to define an I/O compensation cell code for NMOS transistors corresponding to the VDD power rail.
This code is applied to the I/O when the CS bit of SYSCFG_CCCSR is set.
### 12.4.10 SYSCFG register maps

The following table gives the SYSCFG register map and the reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset Register</th>
<th>PC3SO</th>
<th>PC2SO</th>
<th>PC1SO</th>
<th>PC0SO</th>
<th>PA3SO</th>
<th>PA2SO</th>
<th>PA1SO</th>
<th>PA0SO</th>
<th>PB3SO</th>
<th>PB2SO</th>
<th>PB1SO</th>
<th>PB0SO</th>
<th>PB9FM</th>
<th>PB8FM</th>
<th>PB7FM</th>
<th>PB6FM</th>
<th>PB5FM</th>
<th>PB4FM</th>
<th>PB3FM</th>
<th>PB2FM</th>
<th>PB1FM</th>
<th>PB0FM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>SYSCFG_PMCR</td>
<td></td>
<td>PC3SO</td>
<td>PC2SO</td>
<td>PC1SO</td>
<td>PC0SO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>SYSCFG_EXTICR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0C</td>
<td>SYSCFG_EXTICR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x10</td>
<td>SYSCFG_EXTICR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x14</td>
<td>SYSCFG_EXTICR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x18</td>
<td>SYSCFG_CCFGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>SYSCFG_CC CCSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>SYSCFG_CCVR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x28</td>
<td>SYSCFG_CCCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
13 Block interconnect

13.1 Peripheral interconnect

13.1.1 Introduction

Several peripherals have direct connections between them. This enables autonomous communication and synchronization between peripherals, thus saving CPU resources and power consumption. These hardware connections remove software latency, allow the design of a predictable system and result in a reduction of the number of pins and GPIOs.

13.1.2 Connection overview

There are several types of connections.

- Asynchronous connections (A)
  The source output signal is sampled by the destination clock, leading to introduction of a possible jitter in the latency between the source output event and the destination event detection.

- Synchronous connections (S)
  Both source and destination are synchronous (they run on the same clock), and the latency from the source to the destination is deterministic. No jitter is introduced.

- Immediate connections (I)
  Either the source or the destination is an analog signal.

- Break/fault connection for TIM outputs (B)
  The source output signal disables the timer outputs through a pure combinational logic path, without any latency.
Table 75. Peripherals interconnect matrix (CD domain) *(1) (2)*

<table>
<thead>
<tr>
<th>Source</th>
<th>CD domain</th>
<th>SRD domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>APB1</td>
<td>APB2</td>
</tr>
<tr>
<td></td>
<td>TIM2</td>
<td>TIM3</td>
</tr>
<tr>
<td>TIM2</td>
<td>S S</td>
<td>S</td>
</tr>
<tr>
<td>TIM3</td>
<td>S - S S</td>
<td>S</td>
</tr>
<tr>
<td>TIM4</td>
<td>S S - S S</td>
<td>S</td>
</tr>
<tr>
<td>TIM5</td>
<td>- - - S</td>
<td>S - S</td>
</tr>
<tr>
<td>TIM6</td>
<td>- - - S</td>
<td>S</td>
</tr>
<tr>
<td>TIM7</td>
<td>- - - S</td>
<td>S</td>
</tr>
<tr>
<td>TIM13</td>
<td>- - - S</td>
<td>-</td>
</tr>
<tr>
<td>TIM14</td>
<td>- - - S</td>
<td>-</td>
</tr>
<tr>
<td>LPTIM1</td>
<td>- - - A</td>
<td>-</td>
</tr>
<tr>
<td>OPAMP</td>
<td>- - - -</td>
<td>-</td>
</tr>
<tr>
<td>CAN</td>
<td>- - - A</td>
<td>-</td>
</tr>
<tr>
<td>TIM1</td>
<td>S S S S</td>
<td>S</td>
</tr>
<tr>
<td>TIM8</td>
<td>S S S S</td>
<td>S</td>
</tr>
<tr>
<td>TIM15</td>
<td>S S</td>
<td>S</td>
</tr>
<tr>
<td>TIM16</td>
<td>- - - S</td>
<td>S</td>
</tr>
<tr>
<td>TIM17</td>
<td>- - - S</td>
<td>S</td>
</tr>
<tr>
<td>SAI1</td>
<td>A - - -</td>
<td>-</td>
</tr>
<tr>
<td>SAI2</td>
<td>- - - A</td>
<td>-</td>
</tr>
<tr>
<td>DFSDM1</td>
<td>- - - -</td>
<td>-</td>
</tr>
<tr>
<td>ADC1</td>
<td>- - - A</td>
<td>-</td>
</tr>
<tr>
<td>ADC2</td>
<td>- - - A</td>
<td>-</td>
</tr>
<tr>
<td>USB1</td>
<td>A A - -</td>
<td>-</td>
</tr>
<tr>
<td>(OTG_H S1)</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

1. Letters in the table correspond to the type of connection described in Section 13.1.2: Connection overview
2. The "-" symbol in a gray cell means no interconnect.
### Table 76. Peripherals interconnect matrix (SRD domain) \(^{(1)}\) \(^{(2)}\)

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>CD domain</th>
<th>SRD domain</th>
<th>APB4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>APB1</td>
<td>APB2</td>
<td>AHB1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM2</td>
<td>TIM3</td>
<td>TIM4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM5</td>
<td>TIM12</td>
<td>DAC1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CAN</td>
<td>CRS</td>
<td>TIM1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM8</td>
<td>TIM15</td>
<td>TIM16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM17</td>
<td>DFSDM1</td>
<td>ADC1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADC2</td>
<td>DAC2</td>
<td>LPTIM2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM3</td>
<td>DFSDM2</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM1</td>
<td>TRGO</td>
<td>ITR0</td>
</tr>
<tr>
<td></td>
<td>TIM8</td>
<td>TRGO</td>
<td>ITR1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>TIM3</td>
<td>TRGO</td>
<td>ITR2</td>
</tr>
<tr>
<td></td>
<td>TIM4</td>
<td>TRGO</td>
<td>ITR3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AHB1</td>
<td>USB1</td>
<td>SOF</td>
<td>ITR5</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>COMP1</td>
<td>comp1_out</td>
<td>ETR1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>ETR2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RCC</td>
<td>lse_ck</td>
<td>ETR3</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SAI1</td>
<td>SAI1_FS_A</td>
<td>ETR4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAI1</td>
<td>SAI1_FS_B</td>
<td>ETR5</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>COMP1</td>
<td>comp1_out</td>
<td>T14_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>T14_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP1 or</td>
<td>comp1_out or</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2(2)</td>
<td>comp2_out</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>T14_3</td>
<td></td>
</tr>
</tbody>
</table>

1. Letters in the table correspond to the type of connection described in Section 13.1.2: Connection overview.
2. The "-" symbol in a gray cell means no interconnect.

### Table 77. Peripherals interconnect matrix details\(^{(1)}\)

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>Bus</td>
<td>Peripheral</td>
<td>Signal</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM1</td>
<td>TRGO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM8</td>
<td>TRGO</td>
</tr>
<tr>
<td>APB1</td>
<td></td>
<td>TIM3</td>
<td>TRGO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM4</td>
<td>TRGO</td>
</tr>
<tr>
<td>AHB1</td>
<td></td>
<td>USB1</td>
<td>SOF</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>COMP1</td>
<td>comp1_out</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RCC</td>
<td>lse_ck</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SAI1</td>
<td>SAI1_FS_A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAI1</td>
<td>SAI1_FS_B</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>COMP1</td>
<td>comp1_out</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP1 or</td>
<td>comp1_out or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2(2)</td>
<td>comp2_out</td>
</tr>
<tr>
<td>Domain</td>
<td>Bus</td>
<td>Peripheral</td>
<td>Signal</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>------------</td>
<td>--------</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM1</td>
<td>TRGO</td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>TIM2</td>
<td>TRGO</td>
</tr>
<tr>
<td></td>
<td>APB2</td>
<td>TIM15</td>
<td>TRGO</td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>TIM4</td>
<td>TRGO</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>COMP1</td>
<td>comp1_out</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP1</td>
<td>comp1_out</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP1 or COMP2(2)</td>
<td>comp1_out or comp2_out</td>
</tr>
<tr>
<td></td>
<td>APB2</td>
<td>TIM1</td>
<td>TRGO</td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>TIM2</td>
<td>TRGO</td>
</tr>
<tr>
<td></td>
<td>APB2</td>
<td>TIM3</td>
<td>TRGO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CAN</td>
<td>SOC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AHB1</td>
<td>USB1</td>
</tr>
<tr>
<td></td>
<td>APB2</td>
<td>SAI2</td>
<td>SAI2_FS_A</td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>CAN</td>
<td>TMP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CAN</td>
<td>RTP</td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>TIM4</td>
<td>TRGO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM5</td>
<td>TRGO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM12</td>
<td>TRGO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SPDIGRX</td>
<td>FS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AHB1</td>
<td>USB1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SRD</td>
<td>RCC</td>
</tr>
</tbody>
</table>

Table 77. Peripherals interconnect matrix details(1) (continued)
### Table 77. Peripherals interconnect matrix details (continued)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Signal</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Domain</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM15</td>
<td>TRGO</td>
<td>ITR0</td>
<td></td>
<td></td>
<td>CD</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>TIM2</td>
<td>TRGO</td>
<td>ITR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM3</td>
<td>TRGO</td>
<td>ITR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM4</td>
<td>TRGO</td>
<td>ITR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>COMP1</td>
<td>comp1_out</td>
<td>ETR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>ETR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>AHB1</td>
<td>ADC1</td>
<td>adc1_awd1</td>
<td>ETR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADC1</td>
<td>adc1_awd2</td>
<td>ETR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADC1</td>
<td>adc1_awd3</td>
<td>ETR5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>COMP1</td>
<td>comp1_out</td>
<td>TI1_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP1</td>
<td>comp1_out</td>
<td>BRK_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>BRK_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>DFSDM1</td>
<td>dfsdm1_break0</td>
<td>BRK_8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>COMP1</td>
<td>comp1_out</td>
<td>BRK2_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>BRK2_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>DFSDM1</td>
<td>dfsdm1_break1</td>
<td>BRK2_8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>
Table 77. Peripherals interconnect matrix details (1) (continued)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Signal</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM1</td>
<td>TRGO</td>
<td>ITR0</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM2</td>
<td>TRGO</td>
<td>ITR1</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM4</td>
<td>TRGO</td>
<td>ITR2</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM5</td>
<td>TRGO</td>
<td>ITR3</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>COMP1</td>
<td>comp1_out</td>
<td>ETR1</td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>ETR2</td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>AHB1</td>
<td>ADC2</td>
<td>adc2_awd1</td>
<td>ETR3</td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADC2</td>
<td>adc2_awd2</td>
<td>ETR4</td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADC2</td>
<td>adc2_awd3</td>
<td>ETR5</td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>COMP1</td>
<td>comp1_out</td>
<td>BRK_1</td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>BRK_2</td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>DFSDM1</td>
<td>dfsdm1_break2</td>
<td>BRK_8</td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>COMP1</td>
<td>comp1_out</td>
<td>BRK2_1</td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>BRK2_2</td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>DFSDM1</td>
<td>dfsdm1_break3</td>
<td>BRK2_8</td>
<td>B</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 77. Peripherals interconnect matrix details\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Signal</th>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM1</td>
<td>TRGO</td>
<td>ITR0</td>
<td>CD</td>
<td>APB2</td>
<td>TIM16</td>
<td>OC1</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>TIM3</td>
<td>TRGO</td>
<td>ITR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>APB2</td>
<td>TIM16</td>
<td>OC1</td>
<td>ITR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>TIM2</td>
<td>CH1</td>
<td>T11_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM3</td>
<td>CH1</td>
<td>T11_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM4</td>
<td>CH1</td>
<td>T11_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>RCC</td>
<td>lse_ck</td>
<td>T11_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RCC</td>
<td>csi_ck</td>
<td>T11_5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RCC</td>
<td>MCO2</td>
<td>T11_6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM2</td>
<td>CH2</td>
<td>T12_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM3</td>
<td>CH2</td>
<td>T12_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM4</td>
<td>CH2</td>
<td>T12_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>COMP1</td>
<td>comp1_out</td>
<td>BRK_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>BRK_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>DFSDM1</td>
<td>dfsdm_break0</td>
<td>BRK_8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>RCC</td>
<td>lsi_ck</td>
<td>T11_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RCC</td>
<td>lse_ck</td>
<td>T11_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC</td>
<td>WKUP_IT</td>
<td>T11_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP1</td>
<td>comp1_out</td>
<td>BRK_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>BRK_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>DFSDM1</td>
<td>dfsdm_break1</td>
<td>BRK_8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>RCC</td>
<td>HSE_1MHZ</td>
<td>T11_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RCC</td>
<td>MCO1</td>
<td>T11_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP1</td>
<td>comp1_out</td>
<td>BRK_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>BRK_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>DFSDM1</td>
<td>dfsdm_break2</td>
<td>BRK_8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>-</td>
</tr>
</tbody>
</table>
## Table 77. Peripherals interconnect matrix details (continued)

<table>
<thead>
<tr>
<th>Source Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Signal</th>
<th>Destination Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>RTC</td>
<td>rtc_alarm_a_evt</td>
<td>ETR1</td>
<td>LPTIM1</td>
<td>APB1</td>
<td>CD</td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC</td>
<td>rtc_alarm_b_evt</td>
<td>ETR2</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC</td>
<td>rtc_tamp1_evt</td>
<td>ETR3</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC</td>
<td>rtc_tamp2_evt</td>
<td>ETR4</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC</td>
<td>rtc_tamp3_evt</td>
<td>ETR5</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP1</td>
<td>comp1_out</td>
<td>ETR6</td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>ETR7</td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP1</td>
<td>comp1_out</td>
<td>IN1_1</td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>IN2_1</td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC</td>
<td>rtc_alarm_a_evt</td>
<td>ETR1</td>
<td>LPTIM2</td>
<td>APB4</td>
<td>SRD</td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC</td>
<td>rtc_alarm_b_evt</td>
<td>ETR2</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC</td>
<td>rtc_tamp1_evt</td>
<td>ETR3</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC</td>
<td>rtc_tamp2_evt</td>
<td>ETR4</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC</td>
<td>rtc_tamp3_evt</td>
<td>ETR5</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP1</td>
<td>comp1_out</td>
<td>ETR6</td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>ETR7</td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP1</td>
<td>comp1_out</td>
<td>IN1_1</td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>IN1_2</td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP1 or</td>
<td>comp1_out or</td>
<td>IN1_3</td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>IN2_1</td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SRD</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>ETR0</td>
<td>LPTIM3</td>
<td>APB4</td>
<td>SRD</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CD</td>
<td>SAI1</td>
<td>SAI1_FS_A</td>
<td>ETR4</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CD</td>
<td>SAI1</td>
<td>SAI1_FS_B</td>
<td>ETR5</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SRD</td>
<td>DFSDM2</td>
<td>DFSDM2_BRK_0</td>
<td>ETR6</td>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
</tbody>
</table>
Table 77. Peripherals interconnect matrix details\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Signal</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Domain</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM1</td>
<td>TRGO</td>
<td>dac1_ch1/2_</td>
<td>trig1</td>
<td>DAC1</td>
<td>channel 1/channel 2</td>
<td>APB1</td>
<td>CD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM2</td>
<td>TRGO</td>
<td>dac1_ch1/2_</td>
<td>trig2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM4</td>
<td>TRGO</td>
<td>dac1_ch1/2_</td>
<td>trig3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM5</td>
<td>TRGO</td>
<td>dac1_ch1/2_</td>
<td>trig4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM6</td>
<td>TRGO</td>
<td>dac1_ch1/2_</td>
<td>trig5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM7</td>
<td>TRGO</td>
<td>dac1_ch1/2_</td>
<td>trig6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>APB2</td>
<td>TIM8</td>
<td>TRGO</td>
<td>dac1_ch1/2_</td>
<td>trig6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM15</td>
<td>TRGO</td>
<td>dac1_ch1/2_</td>
<td>trig7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>LPTIM1</td>
<td>lptim1_out</td>
<td>dac1_ch1/2_</td>
<td>trig11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>dac1_ch1/2_</td>
<td>trig12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSCFG</td>
<td>EXTI9</td>
<td>dac1_ch1/2_</td>
<td>trig13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SRD</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>dac1_ch1/2_</td>
<td>trig14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^{(1)}\) Table continues on the next page.
### Table 77. Peripherals interconnect matrix details (continued)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Source</th>
<th>Destination</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>APB2</td>
<td>TIM1</td>
<td>TRGO</td>
<td>dac2_ch1_trg1</td>
<td>APB1</td>
<td>TIM2 TRGO</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>TIM2</td>
<td>TRGO</td>
<td>dac2_ch1_trg2</td>
<td>APB1</td>
<td>TIM4 TRGO</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>TIM4</td>
<td>TRGO</td>
<td>dac2_ch1_trg3</td>
<td>APB1</td>
<td>TIM5 TRGO</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>TIM5</td>
<td>TRGO</td>
<td>dac2_ch1_trg4</td>
<td>APB1</td>
<td>TIM6 TRGO</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>TIM6</td>
<td>TRGO</td>
<td>dac2_ch1_trg5</td>
<td>APB1</td>
<td>TIM7 TRGO</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>LPTIM1</td>
<td>lptim1_out dac2_ch1_trg11</td>
<td>APB1</td>
<td>TIM8 TRGO dac2_ch1_trg6</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>TIM15</td>
<td>TRGO</td>
<td>dac2_ch1_trg7</td>
<td>APB1</td>
<td>TIM15 TRGO dac2_ch1_trg7</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APB2</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>dac2_ch1_trg12</td>
<td>APB4</td>
<td>LPTIM2 lptim2_out dac2_ch1_trg12</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>SYSCFG</td>
<td>EXTI9 dac2_ch1_trg13</td>
<td>APB1</td>
<td>SYSCFG EXTI9 dac2_ch1_trg13</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>LPTIM3</td>
<td>lptim3_out</td>
<td>dac2_ch1_trg14</td>
<td>APB1</td>
<td>LPTIM3 lptim3_out dac2_ch1_trg14</td>
<td>S</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 77. Peripherals interconnect matrix details\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Source</th>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Destination</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CD</td>
<td>APB2</td>
<td>TIM1</td>
<td>TRGO</td>
<td>TRG0</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM1</td>
<td>TRG02</td>
<td>TRG1</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM8</td>
<td>TRGO</td>
<td>TRG2</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM8</td>
<td>TRG02</td>
<td>TRG3</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td></td>
<td>TIM3</td>
<td>TRG02</td>
<td>TRG</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM4</td>
<td>TRG02</td>
<td>TRG5</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM16</td>
<td>OC1</td>
<td>TRG6</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td></td>
<td>TIM6</td>
<td>TRGO1</td>
<td>TRG7</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM7</td>
<td>TRGO1</td>
<td>TRG8</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SRD</td>
<td>APB4</td>
<td>SYSCFG</td>
<td>EXTI11</td>
<td>TRG24</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSCFG</td>
<td>EXTI15</td>
<td>TRG25</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>CD</td>
<td>APB1</td>
<td>LPTIM1</td>
<td>lptim1_out</td>
<td>TRG26</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>TRG27</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LPTIM3</td>
<td>lptim3_out</td>
<td>TRG28</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COMP1</td>
<td>comp1_out</td>
<td>TRG29</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>TRG30</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SRD</td>
<td>APB4</td>
<td>SYSCFG</td>
<td>EXTI11</td>
<td>TRG24</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSCFG</td>
<td>EXTI15</td>
<td>TRG25</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>CD</td>
<td>APB1</td>
<td>LPTIM1</td>
<td>lptim1_out</td>
<td>TRG26</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>TRG27</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LPTIM3</td>
<td>lptim3_out</td>
<td>TRG28</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COMP1</td>
<td>comp1_out</td>
<td>TRG29</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COMP2</td>
<td>comp2_out</td>
<td>TRG30</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>APB2</td>
<td></td>
<td>TIM1</td>
<td>OC1</td>
<td>adc_ext_trg0</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM1</td>
<td>OC2</td>
<td>adc_ext_trg1</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM1</td>
<td>OC3</td>
<td>adc_ext_trg2</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM2</td>
<td>OC2</td>
<td>adc_ext_trg3</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM3</td>
<td>TRGO</td>
<td>adc_ext_trg4</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM4</td>
<td>OC4</td>
<td>adc_ext_trg5</td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SRD</td>
<td>APB4</td>
<td>SYSCFG</td>
<td>EXTI11</td>
<td>adc_ext_trg6</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
</tbody>
</table>

\(^{(1)}\) The comments S and A indicate whether the signal is synchronous or asynchronous.
<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Destination</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM8</td>
<td>TRGO</td>
<td>adc_ext_trg7</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM8</td>
<td>TRGO2</td>
<td>adc_ext_trg8</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM1</td>
<td>TRGO</td>
<td>adc_ext_trg9</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM1</td>
<td>TRGO2</td>
<td>adc_ext_trg10</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>TIM2</td>
<td>TRGO</td>
<td>adc_ext_trg11</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM4</td>
<td>TRGO</td>
<td>adc_ext_trg12</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM6</td>
<td>TRGO</td>
<td>adc_ext_trg13</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>APB2</td>
<td>TIM15</td>
<td>TRGO</td>
<td>adc_ext_trg14</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>TIM3</td>
<td>CC4</td>
<td>adc_ext_trg15</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>APB2</td>
<td>LPTIM1</td>
<td>lptim1_out</td>
<td>adc_ext_trg18</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>adc_ext_trg19</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM3</td>
<td>lptim3_out</td>
<td>adc_ext_trg20</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM1</td>
<td>TRGO</td>
<td>adc_jext_trg0</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM1</td>
<td>OC4</td>
<td>adc_jext_trg1</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>TIM2</td>
<td>TRGO</td>
<td>adc_jext_trg2</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM2</td>
<td>OC1</td>
<td>adc_jext_trg3</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM3</td>
<td>OC4</td>
<td>adc_jext_trg4</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM4</td>
<td>TRGO</td>
<td>adc_jext_trg5</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>SYSCFG</td>
<td>EXTI15</td>
<td>adc_jext_trg6</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM8</td>
<td>OC4</td>
<td>adc_jext_trg7</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM1</td>
<td>TRGO2</td>
<td>adc_jext_trg8</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM8</td>
<td>TRGO</td>
<td>adc_jext_trg9</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM8</td>
<td>TRGO2</td>
<td>adc_jext_trg10</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>TIM3</td>
<td>OC3</td>
<td>adc_jext_trg11</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM3</td>
<td>TRGO</td>
<td>adc_jext_trg12</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM3</td>
<td>OC1</td>
<td>adc_jext_trg13</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM6</td>
<td>TRGO</td>
<td>adc_jext_trg14</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>APB2</td>
<td>TIM15</td>
<td>TRGO</td>
<td>adc_jext_trg15</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>APB1</td>
<td>LPTIM1</td>
<td>lptim1_out</td>
<td>adc_jext_trg18</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>adc_jext_trg19</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM3</td>
<td>lptim2_out</td>
<td>adc_jext_trg20</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>
Table 77. Peripherals interconnect matrix details\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Source</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Signal</th>
<th>Destination</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Domain</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM1</td>
<td>OC5</td>
<td>comp_blk1</td>
<td></td>
<td>COMP1 / COMP2</td>
<td>APB4</td>
<td>SRD</td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM2</td>
<td>OC3</td>
<td>comp_blk2</td>
<td></td>
<td>I</td>
<td>-</td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM3</td>
<td>OC3</td>
<td>comp_blk3</td>
<td></td>
<td>I</td>
<td>-</td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM3</td>
<td>OC4</td>
<td>comp_blk4</td>
<td></td>
<td>I</td>
<td>-</td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM8</td>
<td>OC5</td>
<td>comp_blk5</td>
<td></td>
<td>I</td>
<td>-</td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM15</td>
<td>OC1</td>
<td>comp_blk6</td>
<td></td>
<td>I</td>
<td>-</td>
<td></td>
<td></td>
<td>I</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM2</td>
<td>TRGO</td>
<td>SWT0</td>
<td></td>
<td>A</td>
<td>-</td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM3</td>
<td>TRGO</td>
<td>SWT1</td>
<td></td>
<td>A</td>
<td>-</td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM4</td>
<td>TRGO</td>
<td>SWT2</td>
<td></td>
<td>A</td>
<td>-</td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM2</td>
<td>TRGO</td>
<td>EVT0</td>
<td></td>
<td>A</td>
<td>-</td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM3</td>
<td>TRGO</td>
<td>EVT1</td>
<td></td>
<td>A</td>
<td>-</td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM4</td>
<td>TRGO</td>
<td>EVT3</td>
<td></td>
<td>A</td>
<td>-</td>
<td></td>
<td></td>
<td>A</td>
<td>-</td>
</tr>
</tbody>
</table>

1. Letters in the table correspond to the type of connection described in Section 13.1.2: Connection overview.
2. comp1_out and comp2_out are connected to the inputs of an OR gate. The output of this OR gate is connected to the lptim2_in1_mux3 input.
13.2 **Wakeup from low-power modes**

The Extended interrupt and event controller module (EXTI) allows to wake up the system from Stop mode and/or a CPU from CStop mode. Wakeup events are coming from peripherals.

These events are handled by the EXTI either as Configurable events (C), or as Direct events (D). See *Type* column in *Table 78*. Refer to *Section 20: Extended interrupt and event controller (EXTI)* for further details.

Three types of peripheral output signals are connected to the EXTI input events:

- The wake up signals. These signals can be generated by the peripheral without any bus interface clock, they are referred to as xxx_wkup in *Table 78*. Some peripherals do not have this capability.
- The interrupt signals. These signals can be generated only if the peripheral bus interface clock is running. These interrupt signals are generally directly connected to the NVIC of CPU. They are referred to as xxx_it.
- The signals, i.e. the pulses generated by the peripheral. Once a peripheral has generated a signal, no action (flag clearing) is required at peripheral level.

Each EXTI input event has a different wakeup capability or possible target (see *Target* column in *Table 78*):

- CPU wakeup (**CPU**): the input event can be enabled to wake up the CPU
- CPU and SRD domain wakeup for autonomous Run mode (**ANY**): the input event can be enabled to wake up the CPU or the SRD domain only for an autonomous Run mode phase.
### Table 78. EXTI wakeup inputs(1)

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Type</th>
<th>Target</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>Bus</td>
<td>Peripheral</td>
<td>Signal</td>
<td>Signal</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>SYSCFG</td>
<td>exti0_wkup</td>
<td>WKUP0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti1_wkup</td>
<td>WKUP1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti2_wkup</td>
<td>WKUP2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti3_wkup</td>
<td>WKUP3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti4_wkup</td>
<td>WKUP4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti5_wkup</td>
<td>WKUP5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti6_wkup</td>
<td>WKUP6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti7_wkup</td>
<td>WKUP7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti8_wkup</td>
<td>WKUP8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti9_wkup</td>
<td>WKUP9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti10_wkup</td>
<td>WKUP10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti11_wkup</td>
<td>WKUP11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti12_wkup</td>
<td>WKUP12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti13_wkup</td>
<td>WKUP13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti14_wkup</td>
<td>WKUP14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti15_wkup</td>
<td>WKUP15</td>
</tr>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>PWR</td>
<td>pvd_avd_wkup</td>
<td>WKUP16</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>RTC</td>
<td>ALARMS</td>
<td>WKUP17</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>RTC</td>
<td>TAMPER</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>RCC</td>
<td>CSS_LSE</td>
<td>WKUP19</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>RTC</td>
<td>WKUP</td>
<td>WKUP20</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>COMP1</td>
<td>comp1_out</td>
<td>WKUP21</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>I2C1</td>
<td>i2c1_wkup</td>
<td>WKUP22</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>I2C2</td>
<td>i2c2_wkup</td>
<td>WKUP23</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>I2C3</td>
<td>i2c3_wkup</td>
<td>WKUP24</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>I2C4</td>
<td>i2c4_wkup</td>
<td>WKUP25</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>USART1</td>
<td>usart1_wkup</td>
<td>WKUP26</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>USART2</td>
<td>usart2_wkup</td>
<td>WKUP27</td>
</tr>
</tbody>
</table>
### Table 78. EXTI wakeup inputs\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Source</th>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Type</th>
<th>Target</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>APB1</td>
<td>USART3</td>
<td>usart3_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART4</td>
<td>uart4_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART5</td>
<td>WKUP28</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>USART6</td>
<td>usart6_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART7</td>
<td>uart7_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART8</td>
<td>WKUP29</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>UART4</td>
<td>uart4_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART9</td>
<td>uart9_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART10</td>
<td>WKUP30</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SPI1</td>
<td>spi1_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART10</td>
<td>uart10_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART7</td>
<td>WKUP31</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SPI2</td>
<td>spi2_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART5</td>
<td>uart5_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART8</td>
<td>WKUP32</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>SPI3</td>
<td>spi3_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART8</td>
<td>uart8_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART9</td>
<td>WKUP33</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SPI4</td>
<td>spi4_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART8</td>
<td>uart8_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART7</td>
<td>WKUP34</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SPI5</td>
<td>spi5_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>LPTIM1</td>
<td>lptim1_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>LPTIM1</td>
<td>WKUP35</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>MDIOS</td>
<td>mdios_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>MDIOS</td>
<td>mdios_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>LPTIM1</td>
<td>WKUP36</td>
</tr>
<tr>
<td>CD</td>
<td>AHB1</td>
<td>USB1</td>
<td>usb1_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART9</td>
<td>uart9_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART7</td>
<td>WKUP37</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>SPI6</td>
<td>spi6_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART8</td>
<td>uart8_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>UART7</td>
<td>WKUP38</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>LPUART</td>
<td>lpuart_rx_wkup</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_wkup</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>WKUP39</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>LPUART</td>
<td>lpuart_tx_wkup</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM3</td>
<td>WKUP40</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>SPI6</td>
<td>spi6_wkup</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM3</td>
<td>WKUP41</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>SPI6</td>
<td>spi6_wkup</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM3</td>
<td>WKUP42</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>SPI6</td>
<td>spi6_wkup</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM3</td>
<td>WKUP43</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>NC</td>
<td>NC</td>
<td>-</td>
<td>-</td>
<td>NC</td>
<td>NC</td>
<td>-</td>
<td>-</td>
<td>WKUP44</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>LPTIM1</td>
<td>lptim1_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>LPTIM1</td>
<td>lptim1_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>LPTIM1</td>
<td>WKUP45</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_wkup</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_wkup</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>WKUP46</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>WKUP47</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM3</td>
<td>lptim3_wkup</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM3</td>
<td>lptim3_wkup</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM3</td>
<td>WKUP48</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM3</td>
<td>lptim3_out</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM3</td>
<td>lptim3_out</td>
<td>SRD</td>
<td>APB4</td>
<td>LPTIM3</td>
<td>WKUP49</td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>UART9</td>
<td>uart9_wkup</td>
<td>CD</td>
<td>APB2</td>
<td>UART9</td>
<td>uart9_wkup</td>
<td>CD</td>
<td>APB2</td>
<td>UART9</td>
<td>WKUP50</td>
</tr>
<tr>
<td>CD</td>
<td>APB4</td>
<td>USART10</td>
<td>usart10_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>SWPMI</td>
<td>swpmi_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>SWPMI</td>
<td>WKUP51</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>USART10</td>
<td>usart10_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>SWPMI</td>
<td>swpmi_wkup</td>
<td>CD</td>
<td>APB1</td>
<td>SWPMI</td>
<td>WKUP52</td>
</tr>
</tbody>
</table>

**Notes:**
- \(^{(1)}\) EXTI wakeup inputs
- \(^{(2)}\) Additional comment
<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Type</th>
<th>Target</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>PWR</td>
<td>pwr_wkup1_wkup</td>
<td>D</td>
<td>CPU</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>pwr_wkup2_wkup</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>pwr_wkup3_wkup</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>pwr_wkup4_wkup</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>pwr_wkup5_wkup</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>pwr_wkup6_wkup</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>RCC</td>
<td>rcc_it</td>
<td>D</td>
<td>CPU</td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>I2C4</td>
<td>i2c4_ev_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>i2c4_err_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>LPUART1</td>
<td>lpuart1_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>SPI6</td>
<td>spi6_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>BDMA2</td>
<td>bdma2_ch0_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bdma2_ch1_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bdma2_ch2_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bdma2_ch3_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bdma2_ch4_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bdma2_ch5_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bdma2_ch6_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bdma2_ch7_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>DMAMUX2</td>
<td>dmamux2_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>NC</td>
<td>NC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SRD</td>
<td>AHB2</td>
<td>HSEM</td>
<td>hsem_int_it</td>
<td>D</td>
<td>CPU</td>
<td>(1)</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>NC</td>
<td>NC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>NC</td>
<td>NC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>NC</td>
<td>NC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>CEC</td>
<td>cec_wkup</td>
<td>C</td>
<td>CPU</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>NC</td>
<td>NC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>RCC</td>
<td>hse_css_rcc_wkup</td>
<td>D</td>
<td>CPU</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dts_wkup</td>
<td>D</td>
<td>ANY</td>
<td>-</td>
</tr>
</tbody>
</table>

1. The source peripheral needs its bus clock in order to generate the event. The required clock is described in Section Reset and clock controller.

2. The source peripheral signal is not connected to the NVIC.
The Extended Interrupt and Event Controller (EXTI) module event inputs able to wake up the SRD domain for autonomous Run mode have a pending request logic that can be cleared by 4 different input sources (Table 79). Refer to Section 20: Extended interrupt and event controller (EXTI) for further details.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Domain</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>DMAMUX2</td>
<td>dmamux2_evt6</td>
<td>srd_pendclear_in[0]</td>
<td>EXTI</td>
<td>APB4</td>
<td>SRD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux2_evt7</td>
<td>srd_pendclear_in[1]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>srd_pendclear_in[2]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM3</td>
<td>lptim3_out</td>
<td>srd_pendclear_in[3]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
13.3 DMA

In CD domain, the MDMA allows the memory to transfer data. It can be triggered by software or by hardware, according to the connections described in Section 13.3.1.

DMA Multiplexer in CD domain (DMAMUX1) allows to map any peripheral DMA request to any stream of the DMA1 or the DMA2. In addition to this, The DMAMUX provides two other functionalities:

- It’s possible to synchronize a peripheral DMA request with a timer, with an external pin or with a DMA transfer complete of another stream.
- DMA requests can be generated on a stream by the DMAMUX1 itself. This event can be triggered by a timer, by an external pin event, or by a DMA transfer complete of another stream. The number of DMA requests generated is configurable.

The connections on DMAMUX1 and DMA1/DMA2 are described in Section 17: DMA request multiplexer (DMAMUX), Section 15: Direct memory access controller (DMA) and Section 16: Basic direct memory access controller (BDMA).

The BDMA1 in the CD domain is dedicated to the DFSDM1. The connections on BDMA1 are described in Section 13.3.2: DMAMUX1, DMA1, DMA2 and BDMA1 (CD domain).

DMA Multiplexer in SRD domain (DMAMUX2) has the same functionality as DMAMUX1, it is connected to the basic DMA (BDMA2).

The connections on DMAMUX2 and BDMA2 are described in Section 13.3.3: DMAMUX2, BDMA2 (SRD domain). Refer to Section 13.3.3: DMAMUX2, BDMA2 (SRD domain) and Section 16: Basic direct memory access controller (BDMA) for more details.
### 13.3.1 MDMA (CD domain)

Table 80. MDMA

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Source</th>
<th>Destination</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>AHB1</td>
<td>DMA1</td>
<td>dma1_tcif0</td>
<td>dma1_tcif0 mdma_str0</td>
<td>DMA1 stream 0 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma1_tcif1</td>
<td>mdma_str1</td>
<td>DMA1 stream 1 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma1_tcif2</td>
<td>mdma_str2</td>
<td>DMA1 stream 2 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma1_tcif3</td>
<td>mdma_str3</td>
<td>DMA1 stream 3 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma1_tcif4</td>
<td>mdma_str4</td>
<td>DMA1 stream 4 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma1_tcif5</td>
<td>mdma_str5</td>
<td>DMA1 stream 5 transfer complete flag</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma1_tcif6</td>
<td>mdma_str6</td>
<td>DMA1 stream 6 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma1_tcif7</td>
<td>mdma_str7</td>
<td>DMA1 stream 7 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DMA2</td>
<td>dma2_tcif0</td>
<td>mdma_str8</td>
<td>DMA2 stream 0 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma2_tcif1</td>
<td>mdma_str9</td>
<td>DMA2 stream 1 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma2_tcif2</td>
<td>mdma_str10</td>
<td>DMA2 stream 2 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma2_tcif3</td>
<td>mdma_str11</td>
<td>DMA2 stream 3 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma2_tcif4</td>
<td>mdma_str12</td>
<td>DMA2 stream 4 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma2_tcif5</td>
<td>mdma_str13</td>
<td>DMA2 stream 5 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma2_tcif6</td>
<td>mdma_str14</td>
<td>DMA2 stream 6 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dma2_tcif7</td>
<td>mdma_str15</td>
<td>DMA2 stream 7 transfer complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td>APB3</td>
<td>LTDC</td>
<td>ltdc_li_it</td>
<td>mdma_str16</td>
<td>LTDC line interrupt</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Source</th>
<th>Destination</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>APB3</td>
<td>LTDC</td>
<td>ltdc_li_it</td>
<td>mdma_str16</td>
<td>LTDC line interrupt</td>
<td></td>
</tr>
</tbody>
</table>
Table 80. MDMA (continued)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Source</th>
<th>Destination</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>AHB3</td>
<td>JPEG</td>
<td>jpeg_ift_trg</td>
<td>mdma_str17</td>
<td>jpeg input FIFO threshold</td>
</tr>
<tr>
<td>CD</td>
<td>AHB3</td>
<td>OCTOSPI</td>
<td>octospi_ft_trg</td>
<td>mdma_str22</td>
<td>OCTOSPI FIFO threshold</td>
</tr>
<tr>
<td>CD</td>
<td>AHB3</td>
<td>OCTOSPI</td>
<td>octospi_tc_trg</td>
<td>mdma_str23</td>
<td>OCTOSPI FIFO threshold</td>
</tr>
<tr>
<td>CD</td>
<td>AHB3</td>
<td>DMA2D</td>
<td>dma2d_clut_trg</td>
<td>mdma_str24</td>
<td>DMA2D CLUT transfer complete</td>
</tr>
<tr>
<td>CD</td>
<td>AHB3</td>
<td>DMA2D</td>
<td>dma2d_tc_trg</td>
<td>mdma_str25</td>
<td>DMA2D transfer complete</td>
</tr>
<tr>
<td>CD</td>
<td>AHB3</td>
<td>DMA2D</td>
<td>dma2d_tw_trg</td>
<td>mdma_str26</td>
<td>DMA2D transfer watermark</td>
</tr>
<tr>
<td>CD</td>
<td>AHB3</td>
<td>SDMMC1</td>
<td>sdmmc1_dataend_trg</td>
<td>mdma_str29</td>
<td>End of data</td>
</tr>
<tr>
<td>CD</td>
<td>AHB3</td>
<td>SDMMC1</td>
<td>sdmmc1_buffend_trg</td>
<td>mdma_str30</td>
<td>End of buffer</td>
</tr>
<tr>
<td>CD</td>
<td>AHB3</td>
<td>SDMMC1</td>
<td>sdmmc1_cmdend_trg</td>
<td>mdma_str31</td>
<td>End of command</td>
</tr>
<tr>
<td>CD</td>
<td>OCTOSPI2</td>
<td>OCTOSPI2</td>
<td>octospi_ft_trg</td>
<td>mdma_str32</td>
<td>OCTOSPI FIFO threshold</td>
</tr>
<tr>
<td>CD</td>
<td>OCTOSPI2</td>
<td>OCTOSPI2</td>
<td>octospi_fc_trg</td>
<td>mdma_str33</td>
<td>OCTOSPI transfer complete</td>
</tr>
</tbody>
</table>
### 13.3.2 DMAMUX1, DMA1, DMA2 and BDMA1 (CD domain)

**Table 81. DMAMUX1, DMA1, DMA2 and BDMA1 connections**

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Source</th>
<th>Destination</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>dmamux1 internal (Request generator)</td>
<td>dmamux1_req_in1</td>
<td>dmamux1_req_in2</td>
<td>dmamux1_req_in3</td>
<td>dmamux1_req_in4</td>
</tr>
<tr>
<td>CD</td>
<td>AHB1</td>
<td>ADC1</td>
<td>adc1_dma</td>
<td>dmamux1_req_in9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>AHB1</td>
<td>ADC2</td>
<td>adc2_dma</td>
<td>dmamux1_req_in10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM1</td>
<td></td>
<td></td>
<td>DMAMUX1</td>
<td>AHB1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Domain**
- SRD
- CD

**Bus**
- AHB4
- AHB1
- APB2
- APB1

**Peripheral**
- DMAMUX1
- ADC1
- ADC2
- TIM1
- TIM2
- TIM3
- TIM4

**Signal**
- dmamux1_req_in1
- dmamux1_req_in2
- dmamux1_req_in3
- dmamux1_req_in4
- adc1_dma
- adc2_dma
- dmamux1_req_in9
- dmamux1_req_in10
- dmamux1_req_in11
- dmamux1_req_in12
- dmamux1_req_in13
- dmamux1_req_in14
- dmamux1_req_in15
- dmamux1_req_in16
- dmamux1_req_in17
- dmamux1_req_in18
- dmamux1_req_in19
- dmamux1_req_in20
- dmamux1_req_in21
- dmamux1_req_in22
- dmamux1_req_in23
- dmamux1_req_in24
- dmamux1_req_in25
- dmamux1_req_in26
- dmamux1_req_in27
- dmamux1_req_in28
- dmamux1_req_in29
- dmamux1_req_in30
- dmamux1_req_in31
- dmamux1_req_in32

**Comment**
- Requests
### Table 81. DMAMUX1, DMA1, DMA2 and BDMA1 connections (continued)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Signal</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Domain</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>APB1</td>
<td>I2C1</td>
<td>i2c1_rx_dma</td>
<td>dmamux1_req_in33</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>I2C2</td>
<td>i2c2_rx_dma</td>
<td>dmamux1_req_in35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>SPI1</td>
<td>spi1_rx_dma</td>
<td>dmamux1_req_in37</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>SPI2</td>
<td>spi2_rx_dma</td>
<td>dmamux1_req_in39</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>USART1</td>
<td>usart1_rx_dma</td>
<td>dmamux1_req_in41</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>USART2</td>
<td>usart2_rx_dma</td>
<td>dmamux1_req_in43</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>USART3</td>
<td>usart3_rx_dma</td>
<td>dmamux1_req_in45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM8</td>
<td>tim8_ch1_dma</td>
<td>dmamux1_req_in47</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM8</td>
<td>tim8_ch2_dma</td>
<td>dmamux1_req_in48</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM8</td>
<td>tim8_ch3_dma</td>
<td>dmamux1_req_in49</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM8</td>
<td>tim8_ch4_dma</td>
<td>dmamux1_req_in50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM8</td>
<td>tim8_up_dma</td>
<td>dmamux1_req_in51</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM8</td>
<td>tim8_trig_dma</td>
<td>dmamux1_req_in52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM8</td>
<td>tim8_com_dma</td>
<td>dmamux1_req_in53</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>NC</td>
<td>NC</td>
<td>NC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM5</td>
<td>tim5_ch1_dma</td>
<td>dmamux1_req_in55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM5</td>
<td>tim5_ch2_dma</td>
<td>dmamux1_req_in56</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM5</td>
<td>tim5_ch3_dma</td>
<td>dmamux1_req_in57</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM5</td>
<td>tim5_ch4_dma</td>
<td>dmamux1_req_in58</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>SPI3</td>
<td>spi3_rx_dma</td>
<td>dmamux1_req_in61</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>UART4</td>
<td>uart4_rx_dma</td>
<td>dmamux1_req_in63</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>uart4_tx_dma</td>
<td>dmamux1_req_in64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 81. DMAMUX1, DMA1, DMA2 and BDMA1 connections (1) (continued)

<table>
<thead>
<tr>
<th>Source Domain</th>
<th>Source APB</th>
<th>Peripheral</th>
<th>Source Signal</th>
<th>Destination Domain</th>
<th>Destination APB</th>
<th>Destination Signal</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>APB1</td>
<td>UART5</td>
<td>uart5_rx_dma</td>
<td>DMAMUX1</td>
<td>AHB1</td>
<td>dmamux1_req_in65</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>UART5</td>
<td>uart5_tx_dma</td>
<td></td>
<td></td>
<td></td>
<td>Requests</td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>DAC1</td>
<td>dac1_ch1_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>DAC1</td>
<td>dac1_ch2_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM6</td>
<td>tim6_up_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM7</td>
<td>tim7_up_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>USART6</td>
<td>usart6_rx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>USART6</td>
<td>usart6_tx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>I2C3</td>
<td>i2c3_rx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>I2C3</td>
<td>i2c3_tx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>AHB2</td>
<td>DCMI_PSSI</td>
<td>dcmi_dma/pssi_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>AHB2</td>
<td>CRYP</td>
<td>cryp_in_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>AHB2</td>
<td>CRYP</td>
<td>cryp_out_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>AHB2</td>
<td>HASH</td>
<td>hash_in_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>UART7</td>
<td>uart7_rx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>UART7</td>
<td>uart7_tx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>UART8</td>
<td>uart8_rx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>UART8</td>
<td>uart8_tx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SPI4</td>
<td>spi4_rx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SPI4</td>
<td>spi4_tx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SPI5</td>
<td>spi5_rx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SPI5</td>
<td>spi5_tx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SAI1</td>
<td>sai1_a_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SAI1</td>
<td>sai1_b_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SAI2</td>
<td>sai2_a_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>SAI2</td>
<td>sai2_b_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>SWPMI</td>
<td>swpmi_rx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>SWPMI</td>
<td>swpmi_tx_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>SPDIFRX</td>
<td>spdifrx_dt_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>SPDIFRX</td>
<td>spdifrx_cs_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1) Connections may vary depending on the specific hardware configuration.
### Table 81. DMAMUX1, DMA1, DMA2 and BDMA1 connections

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Destination</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Domain</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>APB2</td>
<td>DFSDM1</td>
<td>dfsdm1_dma0</td>
<td>dmamux1_req_in101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dfsdm1_dma1</td>
<td>dmamux1_req_in102</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dfsdm1_dma2</td>
<td>dmamux1_req_in103</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dfsdm1_dma3</td>
<td>dmamux1_req_in104</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM15</td>
<td>tim15_ch1_dma</td>
<td>dmamux1_req_in105</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>tim15_up_dma</td>
<td>dmamux1_req_in106</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>tim15_trig_dma</td>
<td>dmamux1_req_in107</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>tim15_com_dma</td>
<td>dmamux1_req_in108</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM16</td>
<td>tim16_ch1_dma</td>
<td>dmamux1_req_in109</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>tim16_up_dma</td>
<td>dmamux1_req_in110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>TIM17</td>
<td>tim17_ch1_dma</td>
<td>dmamux1_req_in111</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>tim17_up_dma</td>
<td>dmamux1_req_in112</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB2</td>
<td>UART9</td>
<td>uart9_rx_dma</td>
<td>dmamux1_req_in116</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>uart9_tx_dma</td>
<td>dmamux1_req_in117</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>usart10_rx_dma</td>
<td>dmamux1_req_in118</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>usart10_tx_dma</td>
<td>dmamux1_req_in119</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>AHB1</td>
<td>DMAMUX1</td>
<td>dmamux1_evt0</td>
<td>dmamux1_gen0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux1_evt1</td>
<td>dmamux1_gen1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux1_evt2</td>
<td>dmamux1_gen2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>LPTIM1</td>
<td>lptim1_out</td>
<td>dmamux1_gen3</td>
<td></td>
<td></td>
<td></td>
<td>Request generation</td>
</tr>
<tr>
<td>CD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>dmamux1_gen4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td>LPTIM3</td>
<td>lptim3_out</td>
<td>dmamux1_gen5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>EXTI</td>
<td>exti_exti0_it</td>
<td>dmamux1_gen6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM12</td>
<td>tim12_trgo</td>
<td>dmamux1_gen7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>AHB1</td>
<td>DMAMUX1</td>
<td>dmamux1_evt0</td>
<td>dmamux1_trg0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux1_evt1</td>
<td>dmamux1_trg1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux1_evt2</td>
<td>dmamux1_trg2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>LPTIM1</td>
<td>lptim1_out</td>
<td>dmamux1_trg3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB4</td>
<td>LPTIM2</td>
<td>lptim2_out</td>
<td>dmamux1_trg4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td>LPTIM3</td>
<td>lptim3_out</td>
<td>dmamux1_trg5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>EXTI</td>
<td>exti_exti0_it</td>
<td>dmamux1_trg6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>TIM12</td>
<td>tim12_trgo</td>
<td>dmamux1_trg7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes:
- DMAMUX1 AHB1 CD Requests
- DMAMUX1 AHB1 CD Triggers
- DMAMUX1 AHB1 CD Request generation
Table 81. DMAMUX1, DMA1, DMA2 and BDMA1 connections\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Source Domain</th>
<th>Source Bus</th>
<th>Source Peripheral</th>
<th>Source Signal</th>
<th>Destination Domain</th>
<th>Destination Bus</th>
<th>Destination Peripheral</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHB1</td>
<td></td>
<td>DMAMUX1</td>
<td>dmamux1_req_out0</td>
<td>DMA1</td>
<td></td>
<td>dma1_str0</td>
<td>Requests</td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out1</td>
<td></td>
<td></td>
<td>dma1_str2</td>
<td>out</td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out2</td>
<td></td>
<td></td>
<td>dma1_str3</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out3</td>
<td></td>
<td></td>
<td>dma1_str4</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out4</td>
<td></td>
<td></td>
<td>dma1_str5</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out5</td>
<td></td>
<td></td>
<td>dma1_str6</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out6</td>
<td></td>
<td></td>
<td>dma1_str7</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out7</td>
<td></td>
<td></td>
<td>dma2_str0</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out8</td>
<td></td>
<td></td>
<td>dma2_str1</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out9</td>
<td></td>
<td></td>
<td>dma2_str2</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out10</td>
<td></td>
<td></td>
<td>dma2_str3</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out11</td>
<td></td>
<td></td>
<td>dma2_str4</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out12</td>
<td></td>
<td></td>
<td>dma2_str5</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out13</td>
<td></td>
<td></td>
<td>dma2_str6</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td></td>
<td>dmamux1_req_out14</td>
<td></td>
<td></td>
<td>dma2_str7</td>
<td></td>
</tr>
<tr>
<td>APB2</td>
<td></td>
<td>DFSDM1</td>
<td>dfsdm1_dma0</td>
<td>BDMA1</td>
<td></td>
<td>dfsdm1_dma0</td>
<td></td>
</tr>
<tr>
<td>APB2</td>
<td></td>
<td></td>
<td>dfsdm1_dma1</td>
<td></td>
<td></td>
<td>dfsdm1_dma1</td>
<td></td>
</tr>
<tr>
<td>APB2</td>
<td></td>
<td></td>
<td>dfsdm1_dma2</td>
<td></td>
<td></td>
<td>dfsdm1_dma2</td>
<td></td>
</tr>
<tr>
<td>APB2</td>
<td></td>
<td></td>
<td>dfsdm1_dma3</td>
<td></td>
<td></td>
<td>dfsdm1_dma3</td>
<td></td>
</tr>
<tr>
<td>APB2</td>
<td></td>
<td></td>
<td>dfsdm1_dma4</td>
<td></td>
<td></td>
<td>dfsdm1_dma4</td>
<td></td>
</tr>
<tr>
<td>APB2</td>
<td></td>
<td></td>
<td>dfsdm1_dma5</td>
<td></td>
<td></td>
<td>dfsdm1_dma5</td>
<td></td>
</tr>
<tr>
<td>APB2</td>
<td></td>
<td></td>
<td>dfsdm1_dma6</td>
<td></td>
<td></td>
<td>dfsdm1_dma6</td>
<td></td>
</tr>
<tr>
<td>APB2</td>
<td></td>
<td></td>
<td>dfsdm1_dma7</td>
<td></td>
<td></td>
<td>dfsdm1_dma7</td>
<td></td>
</tr>
</tbody>
</table>

1. The “-” symbol in grayed cells means no interconnect.
### 13.3.3 DMAMUX2, BDMA2 (SRD domain)

Table 82. DMAMUX2 and BDMA2 connections

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Source Destination</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>dmamux2 internal (Request generator)</td>
<td>dmamux2_req_in1, dmamux2_req_in2, dmamux2_req_in3, dmamux2_req_in4, dmamux2_req_in5, dmamux2_req_in6, dmamux2_req_in7, dmamux2_req_in8</td>
<td>DMAMUX2 AHB4 SRD Requests</td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>LPUART</td>
<td>dma_rx_lpuart, dma_tx_lpuart</td>
<td>dmamux2_req_in9, dmamux2_req_in10</td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>SPI6</td>
<td>dma_rx_spi6, dma_tx_spi6</td>
<td>dmamux2_req_in11, dmamux2_req_in12</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>APB1</td>
<td>I2C4</td>
<td>dma_rx_i2c4, dma_tx_i2c4</td>
<td>dmamux2_req_in13, dmamux2_req_in14</td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>DAC2, DFSDM2</td>
<td>dma_dac2, dma_dfsdm2</td>
<td>dmamux2_req_in16, dmamux2_req_in17</td>
<td></td>
</tr>
</tbody>
</table>
Table 82. DMAMUX2 and BDMA2 connections (continued)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Source Signal</th>
<th>Destination Signal</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>DMAMUX2</td>
<td>dmamux2_evt0</td>
<td>dmamux2_gen0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux2_evt1</td>
<td>dmamux2_gen1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux2_evt2</td>
<td>dmamux2_gen2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux2_evt3</td>
<td>dmamux2_gen3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux2_evt4</td>
<td>dmamux2_gen4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux2_evt5</td>
<td>dmamux2_gen5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux2_evt6</td>
<td>dmamux2_gen6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>EXTI</td>
<td>exti_lpuart1_rx_it</td>
<td>dmamux2_gen7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti_lpuart1_tx_it</td>
<td>dmamux2_gen8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti_lptim2_wkup</td>
<td>dmamux2_gen9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti_lptim2_out</td>
<td>dmamux2_gen10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti_lptim3_wkup</td>
<td>dmamux2_gen11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti_lptim3_out</td>
<td>dmamux2_gen12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti_i2c4_wkup</td>
<td>dmamux2_gen15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti_spi6_wkup</td>
<td>dmamux2_gen16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti_comp1_out</td>
<td>dmamux2_gen17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti_comp2_out</td>
<td>dmamux2_gen18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti_rtc_wkup</td>
<td>dmamux2_gen19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti_syscfg_exti0</td>
<td>dmamux2_gen20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exti_syscfg_exti2</td>
<td>dmamux2_gen21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>I2C4</td>
<td>it_evt_i2c4</td>
<td>dmamux2_gen22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>SPI6</td>
<td>it_spi6</td>
<td>dmamux2_gen23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>LPUART</td>
<td>it_tx_lpuart1</td>
<td>dmamux2_gen24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>it_rx_lpuart1</td>
<td>dmamux2_gen25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>BDMA2</td>
<td>it_ch0_bdma2</td>
<td>dmamux2_gen28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>it_ch1_bdma2</td>
<td>dmamux2_gen29</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 82. DMAMUX2 and BDMA2 connections (continued)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Bus</th>
<th>Peripheral</th>
<th>Source Signal</th>
<th>Destination Signal</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Domain</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>DMAMUX2</td>
<td>dmamux2_evt0</td>
<td>dmamux2_trg0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux2Evt1</td>
<td>dmamux2Trg1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux2_evt2</td>
<td>dmamux2_trg2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux2_evt3</td>
<td>dmamux2_trg3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux2_evt4</td>
<td>dmamux2_trg4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux2_evt5</td>
<td>dmamux2_trg5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>APB4</td>
<td>EXTI</td>
<td>it_exti_tx_lpuart1</td>
<td>dmamux2_trg6</td>
<td></td>
<td></td>
<td></td>
<td>Triggers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>it_exti_rx_lpuart1</td>
<td>dmamux2_trg7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>it_exti_out_lptim2</td>
<td>dmamux2_trg8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>it_exti_out_lptim3</td>
<td>dmamux2_trg9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>it_exti_wkup_i2c4</td>
<td>dmamux2_trg10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>it_exti_wkup_spi6</td>
<td>dmamux2_trg11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>it_exti_out_comp1</td>
<td>dmamux2_trg12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>it_exti_wkup_RTC</td>
<td>dmamux2_trg13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>it_ext_i_exti0_syscfg</td>
<td>dmamux2_trg14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>it_ext_i_exti2_syscfg</td>
<td>dmamux2_trg15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRD</td>
<td>AHB4</td>
<td>DMAMUX2</td>
<td>dmamux1_req_out0</td>
<td>bdma2_ch0</td>
<td></td>
<td></td>
<td></td>
<td>Requests out</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux1_req_out1</td>
<td>bdma2_ch1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux1_req_out2</td>
<td>bdma2_ch2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux1_req_out3</td>
<td>bdma2_ch3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux1_req_out4</td>
<td>bdma2_ch4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux1_req_out5</td>
<td>bdma2_ch5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux1_req_out6</td>
<td>bdma2_ch6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmamux1_req_out7</td>
<td>bdma2_ch7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
14 MDMA controller (MDMA)

14.1 MDMA introduction

The master direct memory access (MDMA) is used in order to provide high-speed data transfer between memory and memory or between peripherals and memory. Data can be quickly moved by the MDMA without any CPU action. This keeps the CPU resources free for other operations.

The MDMA controller provides a master AXI interface for main memory and peripheral registers access (system access port) and a master AHB interface only for Cortex-M7 TCM memory access (TCM access port).

The MDMA works in conjunction with the standard DMA controllers (DMA1 or DMA2). It offers up to 16 channels, each dedicated to manage memory access requests from one of the DMA stream memory buffer or other peripherals (w/ integrated FIFO).

14.2 MDMA main features

- AXI/AHB master bus architecture, one dedicated to main memory/peripheral accesses and one dedicated to Cortex-M7 AHBS port (only for TCM accesses).
- 16 channels
- Up to 34 hardware trigger sources
- Each channel request can be selected among any of the request sources. This selection is software-configurable and allows several peripherals to initiate DMA requests. The trigger selection can be automatically changed at the end of one block transfer.
- All the channels are identical and can be connected either to a standard DMA or a peripheral request (acknowledge by data read/write) system
- Each channel also supports software trigger
- One 256-level memory buffer, split in two 128-level first-in, first-out (FIFO), that is used to store temporary the data to be transferred (in burst or single transfer mode), for 1 or 2 consecutive buffers. The FIFO stores the data that are transferred during the current channel block transfer (up to the block transfer size). The 2nd FIFO can be used for the next buffer to be transferred, either for the same channel or for the next channel transfer.
- The priorities between the DMA channels are software-programmable (4 levels consisting of very high, high, medium, low) or hardware in case of equality (channel 0 has priority over channel 1, etc.)
- Independent source and destination transfer width (byte, half-word, word, double-word): when the data widths of the source and destination are not equal, the MDMA can pack/unpack the necessary data to optimize the bandwidth.
- The size and address increment for both source and destination can be independently selected.
Note: Based on this separation, some more advanced packing/unpacking operations are available at software level. As an example, 2 x 16-bit data blocks can be interleaved together using two MDMA channels, in the destination memory, by simply programming the 2 channels with an increment step of 4 bytes and a data size of 16-bit + a start address shifted by 2 between the 2 channels.

- Incrementing, decrementing or non incrementing/fixed addressing for source and destination
- Data packing/unpacking is always done respecting the little endian convention: lower address in a data entity (double word, word or half word) contains always the lowest significant byte. This is independent of the address increment/decrement mode of both source and destination.
- Supports incremental burst transfers. The size of the burst is software-configurable, up to 128 bytes. For larger data sizes the burst length is limited, as to respect the maximum 128 bytes data burst size (e.g. 16x64-bit or 32x32-bit).
- For the TCM memory accesses, the burst access is only allowed when the increment and data size are identical and lower than or equal to 32-bit.
- 5 event flags (MDMA Channel Transfer Complete, MDMA Block Transfer complete, MDMA Block Repeat Transfer Complete, MDMA buffer transfer Complete, MDMA Transfer Error) are available and can generate interrupts.
14.3 MDMA functional description

14.3.1 MDMA block diagram

*Figure 80* shows the block diagram of the MDMA.

![MDMA block diagram](image_url)

14.3.2 MDMA internal signals

*Table 83* shows the internal MDMA signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mdma_hclk</td>
<td>Digital input</td>
<td>MDMA AHB clock</td>
</tr>
<tr>
<td>mdma_it</td>
<td>Digital output</td>
<td>MDMA interrupt</td>
</tr>
<tr>
<td>mdma_str[0:31]</td>
<td>Digital input</td>
<td>MDMA stream request</td>
</tr>
</tbody>
</table>
14.3.3 MDMA overview

The MDMA controller performs a direct memory transfer: as an AXI/AHB master, it can take the control of the AXI/AHB bus matrix to initiate AXI/AHB transactions.

It can carry out the following transactions:

- memory-to-memory (software triggered)
- peripheral-to-memory
- memory-to-peripheral

For the last two transaction types, the memory can also be replaced by a memory-mapped peripheral, which has no control over the MDMA flow. When these types of transaction are used and the request is coming from a standard DMA (DMA1 or DMA2), the peripheral register access is replaced by a memory access to the memory buffer used by this DMA.

**Note:** Non-incrementing/decrementing mode is not used for memory accesses.

The source and destination are simply defined by the address (peripherals being memory mapped also).

The AHB slave port is used to program the MDMA controller (it supports 8/16/32-bit accesses).

The size of the data array to be transferred for a single request is one of the following:

1. The buffer transfer size
2. The block size
3. Repeated block
4. Complete channel data (until the linked list pointer for the channel is null)

The choice of the size is done through the TRGM[1:0] (Trigger mode) selection field.

The user must choose one of them based on the data array size available (usually in the DMA1/2 memory buffer) and the “real time” requirements for other MDMA channels (knowing that a buffer transfer is the minimum data aggregate to be transferred by the MDMA without doing a new arbitration between MDMA channel requests).

For each channel, there are three key data array sizes:

1. Burst size: this is the length of the data transfer which can be performed in burst mode. This burst length defines the maximum transfer length which cannot be interrupted at bus arbitration level and can block other masters from accessing the bus.
2. Buffer transfer size: this is the length of the data array to be transferred, on a channel, before checking for MDMA requests on other channels. This is the data array transfer lengths which cannot be interrupted at MDMA level (from other channel requests).
3. Block size: this value has two meanings which can be used together:
   a) main: this is length of the data block which is described in a block structure of the MDMA linked list (corresponds to one entry in the linked list)
   b) selectable: when TRGM[1:0] equals 01, this is the length of the data array which is transferred on a single MDMA request activation (for the respective channel)
14.3.4 MDMA channel

Each of the DMA controller channel provides a unidirectional transfer link between a source and a destination.

Each channel can perform:

- Single block transfer: one block is transferred. At the end of the block, the DMA channel is disabled and an End of Channel Transfer interrupt is generated.
- Repeated block transfer: a number of blocks is transferred before disabling the channel
- Linked list transfer: when the transfer of the current data block (or last block in a repeat) is completed, a new block control structure is loaded from memory and a new block transfer is started.

The minimum amount of data to be transferred for each request (buffer size, up to 128-bytes) is programmable. The total amount of data in a block, is programmable up to 64 Kbytes. This value is decremented after each transfer. When this counter reaches 0, the end of the block is reached and an action is taken based on the repeat counter (for repeated block transfer) and/or linked list structure value.

Note: If the block length is not a multiple of the buffer length, the last buffer transfer in the block is shorter, covering the remaining bytes to be transferred in the current block.

If the link structure address points to a valid memory address, the MDMA reloads the whole channel descriptor structure register contents from memory at this address. Then, a new block transfer is then executed (on the next MDMA channel request) based on this information.

If the link structure address is 0x0, at the end of the current/repeated block transfer, the MDMA channel is disabled and the end of channel transfer interrupt is generated.

14.3.5 Source, destination and transfer modes

Both the source and destination transfers can address peripherals and memories in the entire 4-Gbyte area, at addresses comprised between 0x0000 0000 and 0xFFFF FFFF.

The source/destination addresses can be fixed (e.g. FIFO/single data register peripherals) or incremented/decremented. The transfer can be done in single access or in burst mode (programmable).

14.3.6 Pointer update

The source and destination memory pointers can optionally be automatically post-incremented/decremented or kept constant after each transfer depending on the SINC[1:0] and DINC[1:0] bits in the MDMA_CxCR register.

Disabling the increment mode is useful when the peripheral source or destination data are accessed through a single register/FIFO mode.

If the increment/decrement mode is enabled, the address of the next data transfer is the address of the previous one incremented/decremented by 1, 2, 4 or 8 depending on the increment size programmed in the SINCOS[1:0] or DINCOS[1:0] bits in the MDMA_CxCR register.

In order to optimize the packing operation, the increment offset size and the data size are programmable independently.
14.3.7 MDMA buffer transfer

This is the minimum logical amount of data (up to 128 bytes) which is transferred on an MDMA request event, on one channel.

An MDMA buffer transfer consists of a sequence of a given number of data transfers (done as single or burst data transfers). The number of data items to be transferred and their width (8-bit, 16-bit, 32-bit or 64-bit) are software programmable. The length of the burst used for data transfers is also programmable, independently.

After an event requiring a data array to be transferred, the DMA/peripheral sends a request signal to the MDMA controller. The MDMA controller serves the request depending on the channel priorities.

The request is acknowledged by writing the mask data value to the address given mask address, when these registers are set.

If the mask address register is not set (0x00 value), the request can be reset by simply reading/writing the data to the peripheral. In this case, if the request is done by a destination peripheral, the write must be set as non bufferable, in order to avoid a false new MDMA request.

The total amount of data to be transferred, on the current channel, following a MDMA request, is determined by the TRGM[1:0] field.

If TRGM[1:0] equals 00, a single buffer is transferred, then the MDMA waits for another request on the same channel.

Note: In this case, the hardware request for the currently active channel (data in the FIFO) is not considered again until the end of the write phase for this channel. In this case, even if the channel would still be active at the end of the read phase, another channel (even with lower priority) could start the read phase. Because of this, lower priority channels can be interleaved with current channel transfer.

If TRGM[1:0] is different from 00 (multiple buffers need to be transferred), the mdma_strx for the current channel remains active (internally memorized) until the whole transfer defined by TRGM (block, repeated block or whole channel/linked list data) is completed. However, after transferring an individual buffer, the MDMA enters in a new arbitration phase (between new external requests and internally memorized ones). If no other higher priority, channel request is active, a new buffer transfer is started for the same channel.

Note: When TRGM[1:0] is different from 00, a larger array of data is transferred for a single request. But, as the channel arbitration is done after each buffer transfer, no higher level MDMA requests would be blocked for the more than a buffer transfer period, on any lower priority channel.
14.3.8 Request arbitration

An arbiter manages the MDMA channel requests based on their priority. When MDMA is idle and after the end of each buffer transfer, all MDMA requests (hardware or software) are checked for all enabled channels.

Priorities are managed in two stages:

- Software: each stream priority can be configured in the MDMA_CxCR register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: at hardware level, the channel priority is fixed. If two requests have the same software priority level, the channel with the lower number takes priority over the stream with the higher number. For example, Channel 2 takes priority over Channel 4 when they have the same software priority level.

14.3.9 FIFO

A FIFO structure is used to temporarily store data coming from the source before writing them to the destination. There is a central FIFO structure which is used for all channels.

In order to maximize data bandwidth and bus usage, the following mechanisms are used, allowing multiple read/write operation to be executed in parallel.

- During a buffer transfer, as soon as the FIFO contains enough data for a destination burst transfer, the write operation starts.
- When the complete data for a buffer transfer has been read into the FIFO, the arbitration procedure is started. Following that, the next buffer data to be transferred can be read to the FIFO.

When an active channel is disabled due to an error, during a buffer transfer, the remaining data in the internal FIFO is discarded.
14.3.10 Block transfer
A block is a “contiguous” array of data, up to 64 Kbytes, which is transferred by successive buffer transfers.

Each block of data is defined by the start address and the block length. When a block transfer is completed, one of the following three actions can be executed:
- The block is part of a repeated block transfer: the block length is reloaded and new block start address is computed (based on the information in the MDMA_CxBRUR register)
- It is a single block or the last block in a repeated block transfer: the next block information is loaded from the memory (using the linked list address information, from the MDMA_CxLAR)
- It is the last block which needs to be transferred for the current MDMA channel (MDMA_CxLAR = 0): the channel is disabled and no further MDMA requests are accepted for this channel

14.3.11 Block repeat mode
The block repeat mode allows to repeat a block transfer, with different start addresses for source and destination.

When the repeat block mode is active (repeat counter non 0), at the end of the current block transfer, the block parameters are updated (the BNDT value reloaded and SAR/DAR values updated according to BRSUM/BRDUM configuration), and the repeat counter decremented by 1.

When the repeat block counter reaches 0, this last block is treated as a single block transfer.

14.3.12 Linked list mode
The Linked list mode allows to load a new MDMA configuration (CxTCR, CxBNDTR, CxSAR, CxDAR, CxBRUR, CxLAR, CxTBR, CxMAR and CxMDR registers), from the address given in the CxLAR register. This address must address a memory mapped on the AXI system bus.

Following this operation, the channel is ready to accept new requests, as defined in the block/repeated block modes above, or continue the transfer if TRGM[1:0] equals 11.

The trigger source can be automatically changed, when loading the CxTBR value.

The TRGM and SWRM values must not be changed when TRGM[1:0] equals 11.

14.3.13 MDMA transfer completion
Different events can generate an end of transfer by setting the CTCIF bit in the status register (MDMA_CxISR):
- The MDMA_CxBNDTR counter has reached zero, the Block Repeat Counter is 0 and the Link list pointer address is 0
- The channel is disabled before the end of transfer (by clearing the EN bit in the MDMA_CxCR register) and all the remaining data have been transferred from the FIFO to the destination
14.3.14 MDMA transfer suspension

At any time, a MDMA transfer can be suspended in order to be restarted later on or to be definitively disabled before the end of the MDMA transfer.

There are two cases:

- The channel is disabled, with no later-on restart from the point where it was stopped. There is no particular action to do, besides clearing the EN bit in the MDMA_CxCR register to disable the channel. The stream can take time to be disabled (on going buffer transfer is completed first). The transfer complete interrupt flag is set in order to indicate the end of transfer. The value of the EN bit in MDMA_CxCR is now 0 to confirm the channel interruption. The MDMA_CxNDTR register contains the number of remaining data items at the moment when the channel was stopped so that the software can determine how many data items have been transferred before the channel was interrupted.

- The channel is suspended before the number of remaining bytes to be transferred in the MDMA_CxBNDTR register reaches 0. The aim is to restart the transfer later by re-enabling the channel. The channel transfer complete interrupt flag CTCIF is set in order to indicate the end of transfer. If the MDMA_CxBNDTR, SAR and DAR registers are not modified by software, the transfer continues when the channel is re-enabled. CTCIF must also be reset before restarting the channel.

Note: If the completed buffer is the last of the block, the configuration registers are also updated before disabling the channel, in order to be correctly prepared for a soft restart.

Note: Before reprogramming the channel, software must wait the CTCIF register is set, in order to guarantee that any ongoing operation has been completed.

14.3.15 Error management

The MDMA controller can detect the following errors:

The transfer error interrupt flag (TEIF) is set when:

- A bus error occurs during a MDMA read or a write access
- The address alignment does not correspond to the data size
- The block size is not a multiple of the data size (for source and/or destination): this error is activated on the last transfer and the error address points to the last transfer (which cannot be done)

14.4 MDMA interrupts

For each MDMA channel, an interrupt can be produced on the following events:

- Channel Transfer Completed
- Block-Transfer Completed
- Block-Transfer Repeat Completed
- buffer Transfer Completed
- Transfer Error

Separate interrupt enable control bits are available for flexibility as shown in Table 84.
Note: Before setting an Enable control bit to 1, the corresponding event flag must be cleared, otherwise an interrupt might be immediately generated, if the bit is already set.

Note: When at least one interrupt flag and the respective enable control bit are set, the channel interrupt bit is set in the GISR. The Interrupt output is also activated. This generates an interrupt if the respective interrupt channel is enabled in the NVIC.

14.5 MDMA registers

The MDMA registers can be accessed in word/half-word or byte format.

14.5.1 MDMA global interrupt/status register (MDMA_GISR0)

Address offset: 0x00
Reset value: 0x0000 0000

| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0 | **GIF[15:0]**: Channel x global interrupt flag (x=0..15) |
| This bit is set and reset by hardware. It is a logical OR of all the Channel x interrupt flags (CTCIF, BTIF, BRTIF, TEIF) which are enabled in the interrupt mask register (CTCIEx, BTIEx, BRTIEx, TEIEx) |
| 0: No interrupt generated by channel x |
| 1: Interrupt generated by channel x |

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel Transfer Completed</td>
<td>CTCIF</td>
<td>CTCIE</td>
</tr>
<tr>
<td>Block-Transfer Repeat completed</td>
<td>BTRIF</td>
<td>BTRIE</td>
</tr>
<tr>
<td>Block-Transfer completed</td>
<td>BTIF</td>
<td>BTIE</td>
</tr>
<tr>
<td>Buffer Transfer Completed</td>
<td>TCIF</td>
<td>TCIE</td>
</tr>
<tr>
<td>Transfer Error</td>
<td>TEIF</td>
<td>TEIE</td>
</tr>
</tbody>
</table>

Table 84. MDMA interrupt requests
14.5.2 MDMA channel x interrupt/status register (MDMA_CxISR)

Address offset: 0x40 + 0x40 * x, (x = 0 to 15)
Reset value: 0x0000 0000

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

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

Bit 16 **CRQA**: Channel x ReQuest Active flag
This bit is set by software writing 1 to the SWRQx bit in the MDMA_CxCR register, in order to request a MDMA transfer, and the channel x is enabled.
It is also set by hardware when the channel request become active and the channel is enabled. The hardware request memorized until it is served.
It is cleared by hardware, when the Channel x Request is completed (after the source write phase of the last buffer transfer due for the current request).
0: The MDMA transfer mdma_strx is inactive for channel x.
1: The MDMA transfer mdma_strx is active for channel x
This bit is also reset by hardware when the channel is disabled (in case of transfer error or when reaching the end of the channel data transfer - repeat block = 0 and linked list pointer null - or by software programming the channel enable bit to 0 before that).

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **TCIF**: Channel x buffer transfer complete interrupt flag
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the MDMA_IFCRy register.
0: No buffer transfer complete event on channel x
1: A buffer transfer complete event occurred on channel x
TC is set when a single buffer was transferred. It is activated on each channel transfer request. This can be used as a debug feature (without interrupt), indicating that (at least) an MDMA buffer transfer had been generated since the last flag reset.

Bit 3 **BTIF**: Channel x block transfer complete interrupt flag
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the MDMA_IFCRy register.
0: No block transfer complete event on channel x
1: A block transfer complete event occurred on channel x

Bit 2 **BRTIF**: Channel x block repeat transfer complete interrupt flag
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the MDMA_IFCRy register.
0: No block repeat transfer complete event on channel x
1: A block repeat transfer complete event occurred on channel x
Bit 1 **CTCIF**: Channel x Channel Transfer Complete interrupt flag
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the MDMA_IFCRy register.
0: No channel transfer complete event on channel x
1: A channel transfer complete event occurred on channel x
CTC is set when the last block was transferred and the channel has been automatically disabled.
CTC is also set when the channel is suspended, as a result of writing EN bit to 0.

Bit 0 **TEIF**: Channel x transfer error interrupt flag
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the MDMA_IFCRy register.
0: No transfer error on stream x
1: A transfer error occurred on stream x

### 14.5.3 MDMA channel x interrupt flag clear register (MDMA_CxIFCR)
Address offset: 0x44 + 0x40 * x, (x = 0 to 15)
Reset value: 0x0000 0000

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

Bits 31:5 Reserved, must be kept at reset value.

- **Bit 4 CLTCIF**: Clear buffer Transfer Complete Interrupt Flag for channel x
  Writing 1 into this bit clears TCIF in the MDMA_ISRy register
- **Bit 3 CBTF**: Channel x Clear block transfer complete interrupt flag
  Writing 1 into this bit clears BTIF in the MDMA_ISRy register
- **Bit 2 CBRTIF**: Channel x clear block repeat transfer complete interrupt flag
  Writing 1 into this bit clears BRTIF in the MDMA_ISRy register
- **Bit 1 CTCIF**: Clear Channel transfer complete interrupt flag for channel x
  Writing 1 into this bit clears CTCIF in the MDMA_ISRy register
- **Bit 0 CTEIF**: Channel x clear transfer error interrupt flag
  Writing 1 into this bit clears TEIF in the MDMA_ISRy register
14.5.4 MDMA channel x error status register (MDMA_CxESR)

Address offset: 0x48 + 0x40 * x, (x = 0 to 15)
Reset value: 0x0000 0000

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

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 BSE: Block Size Error
These bits is set by hardware, when the block size is not an integer multiple of the data size either for source or destination. TED indicates whether the problem is on the source or destination.
It is cleared by software writing 1 to the CTEIF bit in the MDMA_IFCRy register.
0: No block size error.
1: Programmed block size is not an integer multiple of the data size.

Bit 10 ASE: Address/Size Error
These bits is set by hardware, when the programmed address is not aligned with the data size. TED indicates whether the problem is on the source or destination.
It is cleared by software writing 1 to the CTEIF bit in the MDMA_IFCRy register.
0: No address/size error.
1: Programmed address is not coherent with the data size.

Bit 9 TEMD: Transfer Error Mask Data
These bits is set by hardware, in case of a transfer error while writing the Mask Data.
It is cleared by software writing 1 to the CTEIF bit in the MDMA_IFCRy register.
0: No mask write access error.
1: The last transfer error on the channel was a related to a write of the Mask Data.

Bit 8 TELD: Transfer Error Link Data
These bits is set by hardware, in case of a transfer error while reading the block link data structure.
It is cleared by software writing 1 to the CTEIF bit in the MDMA_IFCRy register.
0: No link data read access error.
1: The last transfer error on the channel was a related to a read of the Link Data structure.

Bit 7 TED: Transfer Error Direction
These bits is set and cleared by hardware, in case of an MDMA data transfer error.
0: The last transfer error on the channel was a related to a read access.
1: The last transfer error on the channel was a related to a write access.
Bits 6:0  **TEA[6:0]: Transfer Error Address**
These bits are set and cleared by hardware, in case of an MDMA data transfer error. It is used in conjunction with TED.
This field indicates the 7 LSBits of the address which generated a transfer/access error. It can be used by software to retrieve the failing address, by adding this value (truncated to the buffer transfer length size) to the current SAR/DAR value.

*Note:* The SAR/DAR current value doesn’t reflect this last address due to the FIFO management system. The SAR/DAR are only updated at the end of a (buffer) transfer (of TLEN+1 bytes).

*Note:* It is not set in case of a link data error.
14.5.5 MDMA channel x control register (MDMA_CxCR)

This register is used to control the concerned channel.

Address offset: 0x4C + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

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

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

Bit 16 **SWRQ**: Software Request

Writing 1 into this bit sets the CRQA in MDMA_ISRy register, activating the request on Channel x

*Note*: Either the whole CxCR register or the 8-bit/16-bit register at Address offset: 0x4E + 0x40 * channel number can be used for SWRQ activation.

In case of a software request, acknowledge is not generated (neither hardware signal, nor CxMAR write access).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **WEX**: Word Endianess exchange

This bit is set and cleared by software.

0: Little endianess preserved for words
1: word order exchanged in double word

When this bit is set, the word order in the destination double word is reversed: higher address word contains the data read from the lower address of the source.

If destination is not a double word, do not care of the value of this bit.

This bit is protected and can be written only if EN is 0.

Bit 13 **HEX**: Half word Endianess exchange

This bit is set and cleared by software.

0: Little endianess preserved for half words
1: half-word order exchanged in each word

When this bit is set, the half-word order in each destination word is reversed: higher address half-word contains the data read from the lower address of the source.

If destination length is shorter than word, do not care of the value of this bit.

This bit is protected and can be written only if EN is 0.

Bit 12 **BEX**: Byte Endianess exchange

This bit is set and cleared by software.

0: Little endianess preserved for bytes
1: byte order exchanged in each half-word

When this bit is set, the byte order in each destination Half Word is reversed: higher address word contains the data read from the lower address of the source.

If destination is byte, do not care of the value of this bit.

This bit is protected and can be written only if EN is 0.

Bits 11:9 Reserved, must be kept at reset value.
Bit 8  Reserved, must be kept at reset value.

Bits 7:6  **PL[1:0]**: Priority level
These bits are set and cleared by software.
00: Low
01: Medium
10: High
11: Very high
These bits are protected and can be written only if EN is 0.

Bit 5  **TCIE**: buffer Transfer Complete interrupt enable
This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 4  **BTIE**: Block Transfer interrupt enable
This bit is set and cleared by software.
0: BT complete interrupt disabled
1: BT complete interrupt enabled

Bit 3  **BRTIE**: Block Repeat transfer interrupt enable
This bit is set and cleared by software.
0: BT interrupt disabled
1: BT interrupt enabled

Bit 2  **CTCIE**: Channel Transfer Complete interrupt enable
This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 1  **TEIE**: Transfer error interrupt enable
This bit is set and cleared by software.
0: TE interrupt disabled
1: TE interrupt enabled

Bit 0  **EN**: Channel enable / flag channel ready when read low
This bit is set and cleared by software.
0: Channel disabled
1: Channel enabled
This bit can be cleared by hardware:
- on a MDMA end of transfer (stream ready to be configured)
- if a transfer error occurs on the AHB/AXI master buses (bus error/hard fault)
- if another error condition is encountered (data alignment, block/data size incompatibility)

When this bit is reset by software, the ongoing buffer transfer (if any) is completed. All status/configuration registers keep their current values. If the channel is re-enabled without writing these registers, the channel continues from the point where it was interrupted. When this bit is read as 0, the software is allowed to program the configuration registers. It is forbidden to write these registers when the EN bit is read as 1 (writes are ignored).

**Note:** When this bit is reset by software, it is recommended to wait for the CTCIF = 1, in order to ensure that any ongoing buffer transfer has been completed, before reprogramming the channel.
14.5.6 MDMA channel x transfer configuration register (MDMA_CxTCR)

This register is used to configure the concerned channel. In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x00).

Address offset: 0x50 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>BWM: Bufferable Write Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>0: The destination write operation is non-bufferable.</td>
</tr>
<tr>
<td></td>
<td>1: The destination write operation is bufferable.</td>
</tr>
<tr>
<td></td>
<td>This bit is protected and can be written only if EN is 0.</td>
</tr>
<tr>
<td>Note:</td>
<td>All MDMA destination accesses are non-cacheable.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>SWRM: Software Request Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>0: hardware request are taken into account: the transfer is initiated as defined by TRGM value and acknowledged by the MDMA ACKx signal.</td>
</tr>
<tr>
<td></td>
<td>If the CxMAR contains a valid address, the CxMDR value is also written at CxMAR address.</td>
</tr>
<tr>
<td>1: hardware request are ignored. Transfer is triggered by software writing 1 to the SWRQ bit.</td>
<td></td>
</tr>
<tr>
<td>This bit is protected and can be written only if EN is 0.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 29:28</th>
<th>TRGM[1:0]: Trigger Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>These bits are set and cleared by software.</td>
<td></td>
</tr>
<tr>
<td>00: Each MDMA request (software or hardware) triggers a buffer transfer</td>
<td></td>
</tr>
<tr>
<td>01: Each MDMA request (software or hardware) triggers a block transfer</td>
<td></td>
</tr>
<tr>
<td>10: Each MDMA request (software or hardware) triggers a repeated block transfer (if the block repeat is 0, a single block is transferred)</td>
<td></td>
</tr>
<tr>
<td>11: Each MDMA request (software or hardware) triggers the transfer of the whole data for the respective channel (e.g. linked list) until the channel reach the end and it is disabled.</td>
<td></td>
</tr>
<tr>
<td>Note: If TRGM is 11 for the current block, all the values loaded at the end of the current block through the linked list mechanism must keep the same value (TRGM=11) and the same SWRM value, otherwise the result is undefined.</td>
<td></td>
</tr>
<tr>
<td>These bits are protected and can be written only if EN is 0.</td>
<td></td>
</tr>
</tbody>
</table>
Bits 27:26 **PAM[1:0]:** Padding/Alignment mode

These bits are set and cleared by software.

00: Right aligned, padded w/ 0s (default)

*Note:* if source data is larger than destination size, only the LSBs part of the source is written to the destination address. The reminder part is discarded.

01: Right aligned, sign extended

10: Left aligned (padded with 0s)

*Note:* if source data is larger than destination size, only the MSBs part of the source is written to the destination address. The reminder part is discarded.

11: Reserved

When PKE = 1 or DSIZE=SSIZE, these bits are ignored.

These bits are protected and can be written only if EN is 0.

Bit 25 **PKE:** Pack Enable

This bit is set and cleared by software.

0: The source data is written to the destination as is.

If the Source Size is smaller than the destination, it is padded according to the PAM value.

If the Source data size is larger than the destination one, it is truncated. The alignment is done according to the PAM[1:0] value.

1: The source data is packed/unpacked into the destination data size. All data are right aligned, in Little Endian mode.

This bit is protected and can be written only if EN is 0.

Bits 24:18 **TLEN[6:0]:** buffer Transfer Length (number of bytes - 1)

These bits are set and cleared by software.

The value of TLEN+1 represents the number of bytes to be transferred in a single transfer.

The Transfer Length MUST be a multiple of the data size (for both Source and Destination)

*Note:* When the source/destination sizes are different and padding/truncation is used, the TLEN+1 refers to the source data array size.

These bits are protected and can be written only if EN is 0

DBURST value must be programmed in order to ensure that the burst size is lower than the Transfer Size.

Bits 17:15 **DBURST[2:0]:** Destination burst transfer configuration

These bits are set and cleared by software.

000: single transfer

N: burst of $2^N$ beats

These bits are protected and can be written only if EN is 0

DBURST value must be programmed as to ensure that the burst size is lower than the Transfer Length. If this is not ensured, the result is unpredictable.

*Note:* When the destination bus is TCM/AHB (DBUS=1) and DINCOS=11 or DINC=00 or DINCOS/=DSIZE, DBURST must be programmed to 000 (single transfer), else the result is unpredictable.

*Note:* When the destination bus is system/AXI bus (DBUS=0) and DINC=00, DBURST must be maximum 100 (burst of 16), else the result is unpredictable.
Bits 14:12 **SBURST[2:0]**: Source burst transfer configuration
These bits are set and cleared by software.
- 000: single transfer
- N: burst of \(2^N\) beats
These bits are protected and can be written only if EN is 0.
SBURST value must be programmed as to ensure that the burst size is lower than the transfer length. If this is not ensured, the result is unpredictable.
*Note:* When the source bus is TCM (SBUS=1) and SINCOS=11 or SINC = 00 or SINCOS=S_SIZE, SBURST must be programmed to 000 (single transfer), else the result is unpredictable.
*Note:* When the source bus is system/AXI bus (SBUS=0) and SINC=00, SBURST must be maximum 100 (burst of 16), else the result is unpredictable.

Bits 11:10 **DINCOS[1:0]**: Destination increment offset size
These bits are set and cleared by software.
- 00: byte (8-bit)
- 01: half-word (16-bit)
- 10: word (32-bit)
- 11: Double-Word (64-bit)
This bits have no meaning if bit DINC[1:0] = '00'.
These bits are protected and can be written only if EN = '0'.
If DINCOS < DSIZE and DINC /= 00, the result is unpredictable.
If destination is AHB and DBURST /= 000, destination address must be aligned with DINCOS size, else the result is unpredictable.

Bits 9:8 **SINCOS[1:0]**: Source increment offset size
These bits are set and cleared by software.
- 00: byte (8-bit)
- 01: half-word (16-bit)
- 10: word (32-bit)
- 11: Double-Word (64-bit)
This bits have no meaning if bit SINC[1:0] = '00'.
These bits are protected and can be written only if EN = '0'.
If SINCOS < SSIZE and SINC /= 00, the result is unpredictable.
If source is TCM/AHB and SBURST /= 000, source address must be aligned with SINCOS size, else the result is unpredictable.

Bits 7:6 **DSIZE[1:0]**: Destination data size
These bits are set and cleared by software.
- 00: byte (8-bit)
- 01: half-word (16-bit)
- 10: word (32-bit)
- 11: Double-Word (64-bit)
These bits are protected and can be written only if EN is 0.
*Note:* If a value of 11 is programmed for the TCM access/AHB port, a transfer error occurs (TEIF bit set)
If DINCOS < DSIZE and DINC /= 00, the result is unpredictable.
*Note:* DSIZE = 11 (double-word) is forbidden when destination is TCM/AHB bus (DBUS=1).
14.5.7 MDMA channel x block number of data register (MDMA_CxBNDTR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x04).

Address offset: 0x54 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>Bit 15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

**Bits 5:4 SSIZE[1:0]: Source data size**
These bits are set and cleared by software.
00: Byte (8-bit)
01: Half-word (16-bit)
10: Word (32-bit)
11: Double-Word (64-bit)

These bits are protected and can be written only if EN is 0

**Note:** If a value of 11 is programmed for the TCM access/AHB port, a transfer error occurs (TEIF bit set)

**Note:** If SSIZE[1] > SSIZE[0] and SSIZE[0] /= 0, the result is unpredictable.

**Note:** SSIZE = 11 (double-word) is forbidden when source is TCM/AHB bus (SBUS=1).

**Bits 3:2 DINC[1:0]: Destination increment mode**
These bits are set and cleared by software.
00: Destination address pointer is fixed
10: Destination address pointer is incremented after each data transfer (increment is done according to DINCOS)
11: Destination address pointer is decremented after each data transfer (decrement is done according to DINCOS)

These bits are protected and can be written only if EN is 0

**Note:** When destination is AHB (DBUS=1), DINC = 00 is forbidden.

**Bits 1:0 SINC[1:0]: Source increment mode**
These bits are set and cleared by software.
00: Source address pointer is fixed
10: Source address pointer is incremented after each data transfer (increment is done according to SINCOS)
11: Source address pointer is decremented after each data transfer (decrement is done according to SINCOS)

These bits are protected and can be written only if EN is 0

**Note:** When source is AHB (SBUS=1), SINC = 00 is forbidden.
Bits 31:20 **BRC[11:0]: Block Repeat Count**

This field contains the number of repetitions of the current block (0 to 4095). When the channel is enabled, this register is read-only, indicating the remaining number of blocks, excluding the current one. This register decrements after each complete block transfer. Once the last block transfer has completed, this register can either stay at zero or be reloaded automatically from memory (in Linked List mode - i.e. Link Address valid). These bits are protected and can be written only if EN is 0.

Bit 19 **BRDUM: Block Repeat Destination address Update Mode**

0: At the end of a Block transfer, the DAR register is updated by adding the DUV to the current DAR value (current Destination Address)
1: At the end of a block transfer, the DAR register is updated by subtracting the DUV from the current DAR value (current Destination Address)
These bits are protected and can be written only if EN is 0.

Bit 18 **BRSUM: Block Repeat Source address Update Mode**

0: At the end of a block transfer, the SAR register is updated by adding the SUV to the current SAR value (current Source Address)
1: At the end of a block transfer, the SAR register is updated by subtracting the SUV from the current SAR value (current Source Address)
These bits are protected and can be written only if EN is 0.

Bit 17 **Reserved**, must be kept at reset value.

Bits 16:0 **BNDT[16:0]: Block Number of data bytes to transfer**

Number of bytes to be transferred (0 up to 65536) in the current block. When the channel is enabled, this register is read-only, indicating the remaining data items to be transmitted. During the channel activity, this register decrements, indicating the number of data items remaining in the current block. Once the block transfer has completed, this register can either stay at zero or be reloaded automatically with the previously programmed value if the channel is configured in block Repeat mode.

If the value of this register is zero, no transaction can be served even if the stream is enabled.

These bits are protected and can be written only if EN is 0.

**Note:** 1: If the BNDT value is not an integer multiple of the TLEN+1 value, the last transfer is shorter and contain only the remaining data in the Block.

**Note:** 2: The size of the block must be a multiple of the source and destination data size. If this is not true, an error is set and the no data is written.
14.5.8 MDMA channel x source address register (MDMA_CxSAR)

In Linked List mode, at the end of a Block (single or last Block in repeated Block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x08).

Address offset: 0x58 + 0x40 * x, (x = 0 to 15)
Reset value: 0x0000 0000

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

SAR[31:16]

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

SAR[15:0]

Bits 31:0 SAR[31:0]: Source address

These bits represent the base address of the peripheral data register from/to which the data is read. They must be aligned with the SSIZE (e.g. SAR[1:0] = 00 when SSIZE=10), but may be unaligned with the SINCOS.
When source is TCM/AHB, if address is not aligned with SINCOS, access must be programmed as single (SBURST=000).
These bits are write-protected and can be written only when bit EN = ‘0’ in the DMA_SxCR register.
During the channel activity, this register is updated, reflecting the current address from which the data is read next.
When the block repeat mode is active, when a block transfer is completed, the source address is updated by adding/subtracting the SAU value to the current value (already updated after the last transfer in the block).
When the Linked List mode is active, at the end of a block (repeated or not) transfer, the SAR value is loaded from memory (from address LSA + m)

14.5.9 MDMA channel x destination address register (MDMA_CxDAR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x0C).

Address offset: 0x5C + 0x40 * x, (x = 0 to 15)
Reset value: 0x0000 0000

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

DAR[31:16]

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

DAR[15:0]
Bits 31:0 **DAR[31:0]:** Destination address

Base address of the destination address to which the data is written. These bits are write-protected and can be written only when bit EN = '0' in the DMA_SxCR register. Must be aligned with the DSIZE (e.g. DAR[0] = 0 when DSIZE=01), but may be unaligned with the DINCOS.

When destination is AHB, if address is not aligned with DINCOS, access must be programmed as single (DBURST=000).

During the channel activity, this register is updated, reflecting the current address to which the data is written next.

When the block repeat mode is active, when a block transfer is completed, the Destination address is updated by adding/subtracting the DAU value to the current value (after the last transfer in the block).

When the Linked List mode is active, at the end of a block (repeated or not) transfer, the DAR value is loaded from memory (from address LSA + m)
14.5.10 MDMA channel x block repeat address update register (MDMA_CxBRUR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x10).

Address offset: 0x60 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

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

**DUV[15:0]**: Destination address Update Value

This value is used to update (by addition or subtraction) the current destination address at the end of a block transfer. Must be an integer multiple of DSIZE, in order to keep DAR aligned to DSIZE (e.g. DAR[1:0] = 00 when DSIZE=10).

If this value is 0, the next repetition of the block transfer continues to the next address.

When the block repeat mode is not active (BRC=0), this field is ignored.

These bits are write-protected and can be written only when bit EN = '0' in the MDMA_CxCR register.

*Note: This field must be programmed to 0 when DINC[1:0] = 00.*

**SUV[15:0]**: Source address Update Value

This value is used to update (by addition or subtraction) the current source address at the end of a block transfer. Must be an integer multiple of SSIZE, in order to keep SAR aligned to SSIZE (e.g. SAR[1:0] = 00 when SSIZE=10).

If this value is 0, the next repetition of the block transfer continues from the next address.

When the block repeat mode is not active (BRC=0), this field is ignored.

These bits are write-protected and can be written only when bit EN = '0' in the MDMA_CxCR register.

*Note: This field must be programmed to 0 when SINC[1:0] = 00.*
### 14.5.11 MDMA channel x link address register (MDMA_CxLAR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x14).

**Note:** The new value is only taken into account after all registers are updated, for the next end of block.

Address offset: 0x64 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0 LAR[31:0]: Link Address Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the end of a (repeated) block Transfer, the current channel configuration registers (CxTCR, CxBNDTR, CxSAR, CxDAR, CxBRUR, CxMAR, CxMDR and the CxLAR register itself) are loaded with the data structure found at this address.</td>
</tr>
<tr>
<td>If the value of this register is 0, no register update takes place, the channel is disabled and the CTCIF is set, indicating the end of the transfer for this channel. These bits are write-protected and can be written only when bit EN = '0' in the MDMA_CxCR register.</td>
</tr>
<tr>
<td>The channel configuration (LAR address) must be in the AXI address space.</td>
</tr>
<tr>
<td>LAR value must be aligned at a Double Word address, i.e. LAR[2:0] = 0x0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LAR[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LAR[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw r r r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address offset: 0x64 + 0x40 * x, (x = 0 to 15)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value: 0x0000 0000</td>
</tr>
</tbody>
</table>
14.5.12 MDMA channel x trigger and bus selection register (MDMA_CxTBR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x18).

Address offset: 0x68 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:18</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 17</td>
<td><strong>DBUS</strong>: Destination BUS select</td>
</tr>
<tr>
<td>0:</td>
<td>The system/AXI bus is used as destination (write operation) on channel x.</td>
</tr>
<tr>
<td>1:</td>
<td>The AHB bus/TCM is used as destination (write operation) on channel x.</td>
</tr>
<tr>
<td>This bit is protected and can be written only if EN is 0.</td>
<td></td>
</tr>
<tr>
<td>Bit 16</td>
<td><strong>SBUS</strong>: Source BUS select</td>
</tr>
<tr>
<td>0:</td>
<td>The system/AXI bus is used as source (read operation) on channel x.</td>
</tr>
<tr>
<td>1:</td>
<td>The AHB bus/TCM is used as source (read operation) on channel x.</td>
</tr>
<tr>
<td>This bit is protected and can be written only if EN is 0.</td>
<td></td>
</tr>
<tr>
<td>Bits 15:6</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 5:0</td>
<td><strong>TSEL[5:0]</strong>: Trigger Selection</td>
</tr>
<tr>
<td>This bit field selects the hardware trigger (RQ) input for channel x. The ACK is sent on the ACK output having the same index value.</td>
<td></td>
</tr>
<tr>
<td>When SWRM bit is set (software request selected), this bit field is ignored.</td>
<td></td>
</tr>
<tr>
<td>These bits are write-protected and can be written only when bit EN = '0' in the MDMA_CxCR register.</td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong>: If multiple channels are triggered by the same event (have the same TSEL value), all of them are triggered in parallel. However, only the channel with the lowest index acknowledges the request.</td>
<td></td>
</tr>
</tbody>
</table>
14.5.13 MDMA channel x mask address register (MDMA_CxMAR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x20).

Address offset: 0x70 + 0x40 * x, (x = 0 to 15)
Reset value: 0x0000 0000

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

Bits 31:0 MAR[31:0]: Mask address

A write of the MDR value is also done to this address. This allows to clear the RQ signal generated by the DMA2 by writing to its Interrupt Clear register.

If the value of this register is 0, this function is disabled. These bits are write-protected and can be written only when bit EN = '0' in the MDMA_CxCR register.

14.5.14 MDMA channel x mask data register (MDMA_CxMDR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x24).

Address offset: 0x74 + 0x40 * x, (x = 0 to 15)
Reset value: 0x0000 0000

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

Bits 31:0 MDR[31:0]: Mask Data

A write of the MDR value is also done to the address defined by the MAR register. This allows to clear the RQ signal generated by the DMA2 by writing to its Interrupt Clear register.

These bits are write-protected and can be written only when bit EN = '0' in the MDMA_CxCR register.
### 14.5.15 MDMA register map

*Table 85* summarises the MDMA registers.

#### Table 85. MDMA register map and reset values

| Offset  | Register name       | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9   | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|---------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00    | MDMA_GISR0          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04-0x0C| Reserved           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x40    | MDMA_CxISR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x44    | MDMA_CxIFCR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x48    | MDMA_CxESR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4C    | MDMA_CxCR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x50    | MDMA_CxTCR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x54    | MDMA_CxBNDTR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x58    | MDMA_CxSAR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x5C    | MDMA_CxDAR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x60    | MDMA_CxBRUR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x64    | MDMA_CxLAR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x68    | MDMA_CxTBR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x6C    | MDMA_CxMEN         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x70    | MDMA_CxMEN         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x74    | MDMA_CxMDR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Table 85. MDMA register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x74 - 0x7C + 0x40 × channel number</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
15 Direct memory access controller (DMA)

15.1 DMA introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory and between memory and memory. Data can be quickly moved by DMA without any CPU action. This keeps CPU resources free for other operations.

The DMA controller combines a powerful dual AHB master bus architecture with independent FIFO to optimize the bandwidth of the system, based on a complex bus matrix architecture.

The two DMA controllers (DMA1 and DMA2) have 16 streams in total (8 for each controller), each dedicated to managing memory access requests from one or more peripherals.

Each DMA stream is driven by one DMAMUX1 output channel (request). Any DMAMUX1 output request can be individually programmed in order to select the DMA request source signal, from any of the 115 available request input signals.

Refer to the Section 17.3: DMAMUX implementation for more information about the DMA requests and streams mapping.

Each DMA controller has an arbiter for handling the priority between DMA requests.

15.2 DMA main features

The main DMA features are:

- Dual AHB master bus architecture, one dedicated to memory accesses and one dedicated to peripheral accesses
- AHB slave programming interface supporting only 32-bit accesses
- 8 streams for each DMA controller, up to 115 channels (requests) per stream
- Four-word depth 32 first-in, first-out memory buffers (FIFOs) per stream, that can be used in FIFO mode or direct mode:
  - FIFO mode: with threshold level software selectable between 1/4, 1/2 or 3/4 of the FIFO size
  - Direct mode: each DMA request immediately initiates a transfer from/to the memory. When it is configured in direct mode (FIFO disabled), to transfer data in memory-to-peripheral mode, the DMA preloads only one data from the memory to the internal FIFO to ensure an immediate data transfer as soon as a DMA request is triggered by a peripheral.
- Each stream can be configured to be:
  - a regular channel that supports peripheral-to-memory, memory-to-peripheral and memory-to-memory transfers
  - a double buffer channel that also supports double buffering on the memory side
- Priorities between DMA stream requests are software-programmable (4 levels consisting of very high, high, medium, low) or hardware in case of equality (for example, request 0 has priority over request 1)
- Each stream also supports software trigger for memory-to-memory transfers
Each stream request can be selected among up to 115 possible channel requests. This selection is software-configurable by the DMAMUX1 and allows 107 peripherals to initiate DMA requests.

The number of data items to be transferred can be managed either by the DMA controller or by the peripheral:
- DMA flow controller: the number of data items to be transferred is software-programmable from 1 to 65535.
- Peripheral flow controller: the number of data items to be transferred is unknown and controlled by the source or the destination peripheral that signals the end of the transfer by hardware.

Independent source and destination transfer width (byte, half-word, word): when the data widths of the source and destination are not equal, the DMA automatically packs/unpacks the necessary transfers to optimize the bandwidth. This feature is only available in FIFO mode.

Incrementing or non-incrementing addressing for source and destination.

Supports incremental burst transfers of 4, 8 or 16 beats. The size of the burst is software-configurable, usually equal to half the FIFO size of the peripheral.

Each stream supports circular buffer management.

5 event flags (DMA half transfer, DMA transfer complete, DMA transfer error, DMA FIFO error, direct mode error) logically ORed together in a single interrupt request for each stream.
15.3 DMA functional description

15.3.1 DMA block diagram
The figure below shows the block diagram of a DMA.

![DMA block diagram](image)

15.3.2 DMA internal signals
The table below shows the internal DMA signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dma_hclk</td>
<td>Digital input</td>
<td>DMA AHB clock</td>
</tr>
<tr>
<td>dma_it[0:7]</td>
<td>Digital outputs</td>
<td>DMA stream [0:7] global interrupts</td>
</tr>
<tr>
<td>dma_tci[0:7]</td>
<td>Digital outputs</td>
<td>MDMA triggers</td>
</tr>
<tr>
<td>dma:str[0:7]</td>
<td>Digital input</td>
<td>DMA stream [0:7] requests</td>
</tr>
</tbody>
</table>

15.3.3 DMA overview
The DMA controller performs direct memory transfer: as an AHB master, the DMA controller can take the control of the AHB bus matrix to initiate AHB transactions.
The DMA controller carries out the following transactions:

- peripheral-to-memory
- memory-to-peripheral
- memory-to-memory

The DMA controller provides two AHB master ports: the AHB memory port, intended to be connected to memories and the AHB peripheral port, intended to be connected to peripherals. However, to allow memory-to-memory transfers, the AHB peripheral port must also have access to the memories.

The AHB slave port is used to program the DMA controller (it supports only 32-bit accesses).

### 15.3.4 DMA transactions

A DMA transaction consists of a sequence of a given number of data transfers. The number of data items to be transferred and their width (8-bit, 16-bit or 32-bit) are software-programmable.

Each DMA transfer consists of three operations:

- a loading from the peripheral data register or a location in memory, addressed through the DMA_SxPAR or DMA_SxM0AR register
- a storage of the data loaded to the peripheral data register or a location in memory addressed through the DMA_SxPAR or DMA_SxM0AR register
- a post-decrement of the DMA_SxNDTR register, containing the number of transactions that still have to be performed

After an event, the peripheral sends a request signal to the DMA controller. The DMA controller serves the request depending on the channel priorities. As soon as the DMA controller accesses the peripheral, an Acknowledge signal is sent to the peripheral by the DMA controller. The peripheral releases its request as soon as it gets the Acknowledge signal from the DMA controller. Once the request has been deasserted by the peripheral, the DMA controller releases the Acknowledge signal. If there are more requests, the peripheral can initiate the next transaction.

### 15.3.5 DMA request mapping

The DMA request mapping from peripherals to DMA streams is described in *Section 17.3.2: DMAMUX1 mapping.*
15.3.6 Arbiter

An arbiter manages the 8 DMA stream requests based on their priority for each of the two AHB master ports (memory and peripheral ports) and launches the peripheral/memory access sequences.

Priorities are managed in two stages:

- Software: each stream priority can be configured in the DMA_SxCR register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: If two requests have the same software priority level, the stream with the lower number takes priority over the stream with the higher number. For example, stream 2 takes priority over stream 4.

15.3.7 DMA streams

Each of the 8 DMA controller streams provides a unidirectional transfer link between a source and a destination.

Each stream can be configured to perform:

- Regular type transactions: memory-to-peripherals, peripherals-to-memory or memory-to-memory transfers
- Double-buffer type transactions: double buffer transfers using two memory pointers for the memory (while the DMA is reading/writing from/to a buffer, the application can write/read to/from the other buffer).

The amount of data to be transferred (up to 65535) is programmable and related to the source width of the peripheral that requests the DMA transfer connected to the peripheral AHB port. The register that contains the amount of data items to be transferred is decremented after each transaction.

15.3.8 Source, destination and transfer modes

Both source and destination transfers can address peripherals and memories in the entire 4 Gbytes area, at addresses comprised between 0x0000 0000 and 0xFFFF FFFF.

The direction is configured using the DIR[1:0] bits in the DMA_SxCR register and offers three possibilities: memory-to-peripheral, peripheral-to-memory or memory-to-memory transfers.

The table below describes the corresponding source and destination addresses.

<table>
<thead>
<tr>
<th>Bits DIR[1:0] of the DMA_SxCR register</th>
<th>Direction</th>
<th>Source address</th>
<th>Destination address</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Peripheral-to-memory</td>
<td>DMA_SxPAR</td>
<td>DMA_SxM0AR</td>
</tr>
<tr>
<td>01</td>
<td>Memory-to-peripheral</td>
<td>DMA_SxM0AR</td>
<td>DMA_SxPAR</td>
</tr>
</tbody>
</table>
When the data width (programmed in the PSIZE or MSIZE bits in the DMA_SxCR register) is a half-word or a word, respectively, the peripheral or memory address written into the DMA_SxPAR or DMA_SxM0AR/M1AR registers has to be aligned on a word or half-word address boundary, respectively.

**Peripheral-to-memory mode**

Figure 82 describes this mode.

When this mode is enabled (by setting the bit EN in the DMA_SxCR register), each time a peripheral request occurs, the stream initiates a transfer from the source to fill the FIFO.

When the threshold level of the FIFO is reached, the contents of the FIFO are drained and stored into the destination.

The transfer stops once the DMA_SxNDTR register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_SxCR register is cleared by software.

In direct mode (when the DMDIS value in the DMA_SxFCR register is '0'), the threshold level of the FIFO is not used: after each single data transfer from the peripheral to the FIFO, the corresponding data are immediately drained and stored into the destination.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

<table>
<thead>
<tr>
<th>Bits DIR[1:0] of the DMA_SxCR register</th>
<th>Direction</th>
<th>Source address</th>
<th>Destination address</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Memory-to-memory</td>
<td>DMA_SxPAR</td>
<td>DMA_SxM0AR</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Figure 82. Peripheral-to-memory mode

1. For double-buffer mode.

**Memory-to-peripheral mode**

*Figure 83* describes this mode.

When this mode is enabled (by setting the EN bit in the DMA_SxCR register), the stream immediately initiates transfers from the source to entirely fill the FIFO.

Each time a peripheral request occurs, the contents of the FIFO are drained and stored into the destination. When the level of the FIFO is lower than or equal to the predefined threshold level, the FIFO is fully reloaded with data from the memory.

The transfer stops once the DMA_SxNDTR register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_SxCR register is cleared by software.

In direct mode (when the DMDIS value in the DMA_SxFCR register is '0'), the threshold level of the FIFO is not used. Once the stream is enabled, the DMA preloads the first data to transfer into an internal FIFO. As soon as the peripheral requests a data transfer, the DMA transfers the preloaded value into the configured destination. It then reloads again the empty internal FIFO with the next data to be transfer. The preloaded data size corresponds to the value of the PSIZE bitfield in the DMA_SxCR register.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.
1. For double-buffer mode.

**Memory-to-memory mode**

The DMA channels can also work without being triggered by a request from a peripheral. This is the memory-to-memory mode, described in *Figure 84*.

When the stream is enabled by setting the Enable bit (EN) in the DMA_SxCR register, the stream immediately starts to fill the FIFO up to the threshold level. When the threshold level is reached, the FIFO contents are drained and stored into the destination.

The transfer stops once the DMA_SxNDTR register reaches zero or when the EN bit in the DMA_SxCR register is cleared by software.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

---

**Note:** When memory-to-memory mode is used, the circular and direct modes are not allowed.
15.3.9 Pointer incrementation

Peripheral and memory pointers can optionally be automatically post-incremented or kept constant after each transfer depending on the PINC and MINC bits in the DMA_SxCR register.

Disabling the increment mode is useful when the peripheral source or destination data is accessed through a single register.

If the increment mode is enabled, the address of the next transfer is the address of the previous one incremented by 1 (for bytes), 2 (for half-words) or 4 (for words) depending on the data width programmed in the PSIZE or MSIZE bits in the DMA_SxCR register.

In order to optimize the packing operation, it is possible to fix the increment offset size for the peripheral address whatever the size of the data transferred on the AHB peripheral port. The PINCOS bit in the DMA_SxCR register is used to align the increment offset size with the data size on the peripheral AHB port, or on a 32-bit address (the address is then incremented by 4). The PINCOS bit has an impact on the AHB peripheral port only.

If the PINCOS bit is set, the address of the following transfer is the address of the previous one incremented by 4 (automatically aligned on a 32-bit address), whatever the PSIZE value. The AHB memory port, however, is not impacted by this operation.
15.3.10 Circular mode

The circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit in the DMA_SxCR register.

When the circular mode is activated, the number of data items to be transferred is automatically reloaded with the initial value programmed during the stream configuration phase, and the DMA requests continue to be served.

Note: In the circular mode, it is mandatory to respect the following rule in case of a burst mode configured for memory:

\[
\text{DMA}_\text{SxNDTR} = \text{Multiple of } ((\text{Mburst beat}) \times (\text{Msize})/(\text{Psize})), \text{ where:}
\]

- \((\text{Mburst beat}) = 4, 8 \text{ or } 16 \text{ (depending on the MBURST bits in the DMA}_\text{SxCR register)}\)
- \((\text{Msize})/(\text{Psize})) = 1, 2, 4, 1/2 \text{ or } 1/4 \text{ (Msize and Psize represent the MSIZE and PSIZE bits in the DMA}_\text{SxCR register. They are byte dependent)}\)
- \(\text{DMA}_\text{SxNDTR} = \text{Number of data items to transfer on the AHB peripheral port}\)

For example: Mburst beat = 8 (INCR8), MSIZE = ‘00’ (byte) and PSIZE = ‘01’ (half-word), in this case: DMA_SxNDTR must be a multiple of \((8 \times 1/2 = 4)\).

If this formula is not respected, the DMA behavior and data integrity are not guaranteed. NDTR must also be a multiple of the Peripheral burst size multiplied by the peripheral data size, otherwise this could result in a bad DMA behavior.

15.3.11 Double-buffer mode

This mode is available for all the DMA1 and DMA2 streams.

The double-buffer mode is enabled by setting the DBM bit in the DMA_SxCR register.

A double-buffer stream works as a regular (single buffer) stream with the difference that it has two memory pointers. When the double-buffer mode is enabled, the circular mode is automatically enabled (CIRC bit in DMA_SxCR is not relevant) and at each end of transaction, the memory pointers are swapped.

In this mode, the DMA controller swaps from one memory target to another at each end of transaction. This allows the software to process one memory area while the second memory area is being filled/used by the DMA transfer. The double-buffer stream can work in both directions (the memory can be either the source or the destination) as described in Table 88: Source and destination address registers in double-buffer mode (DBM = 1).

Note: In double-buffer mode, it is possible to update the base address for the AHB memory port on-the-fly (DMA_SxM0AR or DMA_SxM1AR) when the stream is enabled, by respecting the following conditions:

- When the CT bit is ‘0’ in the DMA_SxCR register, the DMA_SxM1AR register can be written. Attempting to write to this register while CT = ‘1’ sets an error flag (TEIF) and the stream is automatically disabled.
- When the CT bit is ‘1’ in the DMA_SxCR register, the DMA_SxM0AR register can be written. Attempting to write to this register while CT = ‘0’, sets an error flag (TEIF) and the stream is automatically disabled.

To avoid any error condition, it is advised to change the base address as soon as the TCIF flag is asserted because, at this point, the targeted memory must have changed from...
memory 0 to 1 (or from 1 to 0) depending on the value of CT in the DMA_SxCR register in accordance with one of the two above conditions.

For all the other modes (except the double-buffer mode), the memory address registers are write-protected as soon as the stream is enabled.

Table 88. Source and destination address registers in double-buffer mode (DBM = 1)

<table>
<thead>
<tr>
<th>Bits DIR[1:0] of the DMA_SxCR register</th>
<th>Direction</th>
<th>Source address</th>
<th>Destination address</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Peripheral-to-memory</td>
<td>DMA_SxPAR</td>
<td>DMA_SxM0AR / DMA_SxM1AR</td>
</tr>
<tr>
<td>01</td>
<td>Memory-to-peripheral</td>
<td>DMA_SxM0AR / DMA_SxM1AR</td>
<td>DMA_SxPAR</td>
</tr>
<tr>
<td>10</td>
<td>Not allowed(1)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. When the double-buffer mode is enabled, the circular mode is automatically enabled. Since the memory-to-memory mode is not compatible with the circular mode, when the double-buffer mode is enabled, it is not allowed to configure the memory-to-memory mode.

15.3.12 Programmable data width, packing/unpacking, endianness

The number of data items to be transferred has to be programmed into DMA_SxNDTR (number of data items to transfer bit, NDT) before enabling the stream (except when the flow controller is the peripheral, PFCTRL bit in DMA_SxCR is set).

When using the internal FIFO, the data widths of the source and destination data are programmable through the PSIZE and MSIZE bits in the DMA_SxCR register (can be 8-, 16- or 32-bit).

When PSIZE and MSIZE are not equal:

- The data width of the number of data items to transfer, configured in the DMA_SxNDTR register is equal to the width of the peripheral bus (configured by the PSIZE bits in the DMA_SxCR register). For instance, in case of peripheral-to-memory, memory-to-peripheral or memory-to-memory transfers and if the PSIZE[1:0] bits are configured for half-word, the number of bytes to be transferred is equal to 2 × NDT.

- The DMA controller only copes with little-endian addressing for both source and destination. This is described in Table 89: Packing/unpacking and endian behavior (bit PINC = MINC = 1).

This packing/unpacking procedure may present a risk of data corruption when the operation is interrupted before the data are completely packed/unpacked. So, to ensure data coherence, the stream may be configured to generate burst transfers: in this case, each group of transfers belonging to a burst are indivisible (refer to Section 15.3.13: Single and burst transfers).

In direct mode (DMDIS = 0 in the DMA_SxFCR register), the packing/unpacking of data is not possible. In this case, it is not allowed to have different source and destination transfer data widths: both are equal and defined by the PSIZE bits in the DMA_SxCR register. MSIZE bits are not relevant.
Table 89. Packing/unpacking and endian behavior (bit PINC = MINC = 1)

<table>
<thead>
<tr>
<th>AHB memory port width</th>
<th>AHB peripheral port width</th>
<th>Number of data items to transfer (NDT)</th>
<th>Memory transfer number</th>
<th>Memory port address / byte lane</th>
<th>Peripheral transfer number</th>
<th>Peripheral port address / byte lane</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
<td>4</td>
<td>1</td>
<td>0x0 / 80[7:0]</td>
<td>1</td>
<td>0x0 / 80[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1 / 81[7:0]</td>
<td>2</td>
<td>0x1 / 81[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x2 / 82[7:0]</td>
<td>3</td>
<td>0x2 / 82[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3 / 83[7:0]</td>
<td>4</td>
<td>0x3 / 83[7:0]</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>2</td>
<td>1</td>
<td>0x0 / 80[7:0]</td>
<td>1</td>
<td>0x0 / 80[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1 / 81[7:0]</td>
<td>2</td>
<td>0x1 / 81[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x2 / 82[7:0]</td>
<td>3</td>
<td>0x2 / 82[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3 / 83[7:0]</td>
<td>4</td>
<td>0x3 / 83[7:0]</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>0x0 / 80[7:0]</td>
<td>1</td>
<td>0x0 / 80[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1 / 81[7:0]</td>
<td>2</td>
<td>0x1 / 81[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x2 / 82[7:0]</td>
<td>3</td>
<td>0x2 / 82[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3 / 83[7:0]</td>
<td>4</td>
<td>0x3 / 83[7:0]</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>4</td>
<td>1</td>
<td>0x0 / 160[15:0]</td>
<td>1</td>
<td>0x0 / 160[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1 / 161[15:0]</td>
<td>2</td>
<td>0x1 / 161[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x2 / 162[15:0]</td>
<td>3</td>
<td>0x2 / 162[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3 / 163[15:0]</td>
<td>4</td>
<td>0x3 / 163[15:0]</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>2</td>
<td>1</td>
<td>0x0 / 160[15:0]</td>
<td>1</td>
<td>0x0 / 160[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1 / 161[15:0]</td>
<td>2</td>
<td>0x1 / 161[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x2 / 162[15:0]</td>
<td>3</td>
<td>0x2 / 162[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3 / 163[15:0]</td>
<td>4</td>
<td>0x3 / 163[15:0]</td>
</tr>
<tr>
<td>16</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>0x0 / 160[15:0]</td>
<td>1</td>
<td>0x0 / 160[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1 / 161[15:0]</td>
<td>2</td>
<td>0x1 / 161[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x2 / 162[15:0]</td>
<td>3</td>
<td>0x2 / 162[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3 / 163[15:0]</td>
<td>4</td>
<td>0x3 / 163[15:0]</td>
</tr>
<tr>
<td>32</td>
<td>8</td>
<td>4</td>
<td>1</td>
<td>0x0 / 320[31:0]</td>
<td>1</td>
<td>0x0 / 320[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1 / 321[31:0]</td>
<td>2</td>
<td>0x1 / 321[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x2 / 322[31:0]</td>
<td>3</td>
<td>0x2 / 322[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3 / 323[31:0]</td>
<td>4</td>
<td>0x3 / 323[31:0]</td>
</tr>
<tr>
<td>32</td>
<td>16</td>
<td>2</td>
<td>1</td>
<td>0x0 / 320[31:0]</td>
<td>1</td>
<td>0x0 / 320[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1 / 321[31:0]</td>
<td>2</td>
<td>0x1 / 321[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x2 / 322[31:0]</td>
<td>3</td>
<td>0x2 / 322[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3 / 323[31:0]</td>
<td>4</td>
<td>0x3 / 323[31:0]</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>0x0 / 320[31:0]</td>
<td>1</td>
<td>0x0 / 320[31:0]</td>
</tr>
</tbody>
</table>

Table 90. Restriction on NDT versus PSIZE and MSIZE

<table>
<thead>
<tr>
<th>PSIZE[1:0] of DMA_SxCR</th>
<th>MSIZE[1:0] of DMA_SxCR</th>
<th>NDT[15:0] of DMA_SxNDTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 (8-bit)</td>
<td>01 (16-bit)</td>
<td>Must be a multiple of 2.</td>
</tr>
<tr>
<td>00 (8-bit)</td>
<td>10 (32-bit)</td>
<td>Must be a multiple of 4.</td>
</tr>
<tr>
<td>01 (16-bit)</td>
<td>10 (32-bit)</td>
<td>Must be a multiple of 4.</td>
</tr>
</tbody>
</table>

Note: Peripheral port may be the source or the destination (it could also be the memory source in the case of memory-to-memory transfer).

PSIZE, MSIZE and NDT[15:0] must be configured so as to ensure that the last transfer is not incomplete. This can occur when the data width of the peripheral port (PSIZE bits) is lower than the data width of the memory port (MSIZE bits). This constraint is summarized in the table below.
15.3.13 Single and burst transfers

The DMA controller can generate single transfers or incremental burst transfers of 4, 8 or 16 beats.

The size of the burst is configured by software independently for the two AHB ports by using the MBURST[1:0] and PBURST[1:0] bits in the DMA_SxCR register.

The burst size indicates the number of beats in the burst, not the number of bytes transferred.

To ensure data coherence, each group of transfers that form a burst are indivisible: AHB transfers are locked and the arbiter of the AHB bus matrix does not degrant the DMA master during the sequence of the burst transfer.

Depending on the single or burst configuration, each DMA request initiates a different number of transfers on the AHB peripheral port:

- When the AHB peripheral port is configured for single transfers, each DMA request generates a data transfer of a byte, half-word or word depending on the PSIZE[1:0] bits in the DMA_SxCR register.
- When the AHB peripheral port is configured for burst transfers, each DMA request generates 4, 8 or 16 beats of byte, half word or word transfers depending on the PBURST[1:0] and PSIZE[1:0] bits in the DMA_SxCR register.

The same as above has to be considered for the AHB memory port considering the MBURST and MSIZE bits.

In direct mode, the stream can only generate single transfers and the MBURST[1:0] and PBURST[1:0] bits are forced by hardware.

The address pointers (DMA_SxPAR or DMA_SxM0AR registers) must be chosen so as to ensure that all transfers within a burst block are aligned on the address boundary equal to the size of the transfer.

The burst configuration has to be selected in order to respect the AHB protocol, where bursts must not cross the 1 Kbyte address boundary because the minimum address space that can be allocated to a single slave is 1 Kbyte. This means that the 1-Kbyte address boundary must not be crossed by a burst block transfer, otherwise an AHB error is generated, that is not reported by the DMA registers.

15.3.14 FIFO

FIFO structure

The FIFO is used to temporarily store data coming from the source before transmitting them to the destination.

Each stream has an independent 4-word FIFO and the threshold level is software-configurable between 1/4, 1/2, 3/4 or full.

To enable the use of the FIFO threshold level, the direct mode must be disabled by setting the DMDIS bit in the DMA_SxFCR register.
The structure of the FIFO differs depending on the source and destination data widths, and is described in the figure below.

**Figure 85. FIFO structure**

- **Source: byte**
  - B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
  - B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0

- **Destination: word**
  - W3, W2, W1, W0

- **Source: byte**
  - H7 H6 H5 H4 H3 H2 H1 H0
  - H7 H6 H5 H4 H3 H2 H1 H0

- **Destination: half-word**
  - H7, H6, H5, H4, H3, H2, H1, H0

- **Source: half-word**
  - H7 H6 H5 H4 H3 H2 H1 H0
  - H7 H6 H5 H4 H3 H2 H1 H0

- **Destination: word**
  - W3, W2, W1, W0

- **Source: half-word**
  - H7 H6 H5 H4 H3 H2 H1 H0
  - H7 H6 H5 H4 H3 H2 H1 H0

- **Destination: byte**
  - B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
  - B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0

- **Destination: byte**
  - B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
  - B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
FIFO threshold and burst configuration

Caution is required when choosing the FIFO threshold (bits FTH[1:0] of the DMA_SxFCR register) and the size of the memory burst (MBURST[1:0] of the DMA_SxCR register): The content pointed by the FIFO threshold must exactly match an integer number of memory burst transfers. If this is not the case, a FIFO error (flag FEIFx of the DMA_HISR or DMA_LISR register) is generated when the stream is enabled, then the stream is automatically disabled. The allowed and forbidden configurations are described in the table below. The forbidden configurations are highlighted in gray in the table.

In all cases, the burst size multiplied by the data size must not exceed the FIFO size (data size can be: 1 (byte), 2 (half-word) or 4 (word)).

Incomplete burst transfer at the end of a DMA transfer may happen if one of the following conditions occurs:

- For the AHB peripheral port configuration: the total number of data items (set in the DMA_SxNDTR register) is not a multiple of the burst size multiplied by the data size.
- For the AHB memory port configuration: the number of remaining data items in the FIFO to be transferred to the memory is not a multiple of the burst size multiplied by the data size.

In such cases, the remaining data to be transferred is managed in single mode by the DMA, even if a burst transaction is requested during the DMA stream configuration.

Note: When burst transfers are requested on the peripheral AHB port and the FIFO is used (DMDIS = 1 in the DMA_SxCR register), it is mandatory to respect the following rule to avoid permanent underrun or overrun conditions, depending on the DMA stream direction:

\[ \text{If } (\text{PBURST} \times \text{PSIZE}) = \text{FIFO_SIZE} \text{ (4 words), FIFO_Threshold = 3/4 is forbidden with PSIZE = 1, 2 or 4 and PBURST = 4, 8 or 16.} \]

This rule ensures that enough FIFO space at a time is free to serve the request from the peripheral.

### Table 91. FIFO threshold configurations

<table>
<thead>
<tr>
<th>MSIZE</th>
<th>FIFO level</th>
<th>MBURST = INCR4</th>
<th>MBURST = INCR8</th>
<th>MBURST = INCR16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>1/4</td>
<td>1 burst of 4 beats</td>
<td>Forbidden</td>
<td>Forbidden</td>
</tr>
<tr>
<td></td>
<td>1/2</td>
<td>2 bursts of 4 beats</td>
<td>1 burst of 8 beats</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3/4</td>
<td>3 bursts of 4 beats</td>
<td>Forbidden</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Full</td>
<td>4 bursts of 4 beats</td>
<td>2 bursts of 8 beats</td>
<td>1 burst of 16 beats</td>
</tr>
<tr>
<td>Half-word</td>
<td>1/4</td>
<td>Forbidden</td>
<td>Forbidden</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1/2</td>
<td>1 burst of 4 beats</td>
<td>Forbidden</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3/4</td>
<td>Forbidden</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Full</td>
<td>2 bursts of 4 beats</td>
<td>1 burst of 8 beats</td>
<td>Forbidden</td>
</tr>
<tr>
<td>Word</td>
<td>1/4</td>
<td>Forbidden</td>
<td>Forbidden</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1/2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3/4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Full</td>
<td>1 burst of 4 beats</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In all cases, the burst size multiplied by the data size must not exceed the FIFO size (data size can be: 1 (byte), 2 (half-word) or 4 (word)).
FIFO flush

The FIFO can be flushed when the stream is disabled by resetting the EN bit in the DMA_SxCR register and when the stream is configured to manage peripheral-to-memory or memory-to-memory transfers. If some data are still present in the FIFO when the stream is disabled, the DMA controller continues transferring the remaining data to the destination (even though stream is effectively disabled). When this flush is completed, the transfer complete status bit (TCIFx) in the DMA_LISR or DMA_HISR register is set.

The remaining data counter DMA_SxNDTR keeps the value in this case to indicate how many data items are currently available in the destination memory.

Note that during the FIFO flush operation, if the number of remaining data items in the FIFO to be transferred to memory (in bytes) is less than the memory data width (for example 2 bytes in FIFO while MSIZE is configured to word), data is sent with the data width set in the MSIZE bit in the DMA_SxCR register. This means that memory is written with an undesired value. The software may read the DMA_SxNDTR register to determine the memory area that contains the good data (start address and last address).

If the number of remaining data items in the FIFO is lower than a burst size (if the MBURST bits in DMA_SxCR register are set to configure the stream to manage burst on the AHB memory port), single transactions are generated to complete the FIFO flush.

Direct mode

By default, the FIFO operates in direct mode (DMDIS bit in the DMA_SxFCR is reset) and the FIFO threshold level is not used. This mode is useful when the system requires an immediate and single transfer to or from the memory after each DMA request.

When the DMA is configured in direct mode (FIFO disabled), to transfer data in memory-to-peripheral mode, the DMA preloads one data from the memory to the internal FIFO to ensure an immediate data transfer as soon as a DMA request is triggered by a peripheral.

To avoid saturating the FIFO, it is recommended to configure the corresponding stream with a high priority.

This mode is restricted to transfers where:

- the source and destination transfer widths are equal and both defined by the PSIZE[1:0] bits in DMA_SxCR (MSIZE[1:0] bits are not relevant)
- burst transfers are not possible (PBURST[1:0] and MBURST[1:0] bits in DMA_SxCR are don’t care)

Direct mode must not be used when implementing memory-to-memory transfers.

15.3.15 DMA transfer completion

Different events can generate an end of transfer by setting the TCIFx bit in the DMA_LISR or DMA_HISR status register:

- In DMA flow controller mode:
  - The DMA_SxNDTR counter has reached zero in the memory-to-peripheral mode.
  - The stream is disabled before the end of transfer (by clearing the EN bit in the DMA_SxCR register) and (when transfers are peripheral-to-memory or memory-
to-memory) all the remaining data have been flushed from the FIFO into the memory.

- In Peripheral flow controller mode:
  - The last external burst or single request has been generated from the peripheral and (when the DMA is operating in peripheral-to-memory mode) the remaining data have been transferred from the FIFO into the memory
  - The stream is disabled by software, and (when the DMA is operating in peripheral-to-memory mode) the remaining data have been transferred from the FIFO into the memory

**Note:** The transfer completion is dependent on the remaining data in FIFO to be transferred into memory only in the case of peripheral-to-memory mode. This condition is not applicable in memory-to-peripheral mode.

If the stream is configured in noncircular mode, after the end of the transfer (that is when the number of data to be transferred reaches zero), the DMA is stopped (EN bit in DMA_SxCR register is cleared by Hardware) and no DMA request is served unless the software reprograms the stream and re-enables it (by setting the EN bit in the DMA_SxCR register).

### 15.3.16 DMA transfer suspension

At any time, a DMA transfer can be suspended to be restarted later on or to be definitively disabled before the end of the DMA transfer.

There are two cases:

- The stream disables the transfer with no later-on restart from the point where it was stopped. There is no particular action to do, except to clear the EN bit in the DMA_SxCR register to disable the stream. The stream may take time to be disabled (ongoing transfer is completed first). The transfer complete interrupt flag (TCIF in the DMA_LISR or DMA_HISR register) is set in order to indicate the end of transfer. The value of the EN bit in DMA_SxCR is now ‘0’ to confirm the stream interruption. The DMA_SxNDTR register contains the number of remaining data items at the moment when the stream was stopped so that the software can determine how many data items have been transferred before the stream was interrupted.

- The stream suspends the transfer before the number of remaining data items to be transferred in the DMA_SxNDTR register reaches 0. The aim is to restart the transfer later by re-enabling the stream. In order to restart from the point where the transfer was stopped, the software has to read the DMA_SxNDTR register after disabling the stream by writing the EN bit in DMA_SxCR register (and then checking that it is at ‘0’) to know the number of data items already collected. Then:
  - The peripheral and/or memory addresses have to be updated in order to adjust the address pointers
  - The SxNDTR register has to be updated with the remaining number of data items to be transferred (the value read when the stream was disabled)
  - The stream may then be re-enabled to restart the transfer from the point it was stopped

**Note:** A transfer complete interrupt flag (TCIF in DMA_LISR or DMA_HISR) is set to indicate the end of transfer due to the stream interruption.
15.3.17 Flow controller

The entity that controls the number of data to be transferred is known as the flow controller. This flow controller is configured independently for each stream using the PFCTRL bit in the DMA_SxCR register.

The flow controller can be:

- The DMA controller: in this case, the number of data items to be transferred is programmed by software into the DMA_SxNDTR register before the DMA stream is enabled.
- The peripheral source or destination: this is the case when the number of data items to be transferred is unknown. The peripheral indicates by hardware to the DMA controller when the last data are being transferred. This feature is only supported for peripherals that are able to signal the end of the transfer.

When the peripheral flow controller is used for a given stream, the value written into the DMA_SxNDTR has no effect on the DMA transfer. Actually, whatever the value written, it is forced by hardware to 0xFFFF as soon as the stream is enabled, to respect the following schemes:

- Anticipated stream interruption: EN bit in DMA_SxCR register is reset to 0 by the software to stop the stream before the last data hardware signal (single or burst) is sent by the peripheral. In such a case, the stream is switched off and the FIFO flush is triggered in the case of a peripheral-to-memory DMA transfer. The TCIFx flag of the corresponding stream is set in the status register to indicate the DMA completion. To know the number of data items transferred during the DMA transfer, read the DMA_SxNDTR register and apply the following formula:
  \[ \text{Number of data transferred} = 0xFFFF - \text{DMA_SxNDTR} \]

- Normal stream interruption due to the reception of a last data hardware signal: the stream is automatically interrupted when the peripheral requests the last transfer (single or burst) and when this transfer is complete. The TCIFx flag of the corresponding stream is set in the status register to indicate the DMA transfer completion. To know the number of data items transferred, read the DMA_SxNDTR register and apply the same formula as above.

- The DMA_SxNDTR register reaches 0: the TCIFx flag of the corresponding stream is set in the status register to indicate the forced DMA transfer completion. The stream is automatically switched off even though the last data hardware signal (single or burst) has not been yet asserted. The already transferred data is not lost. This means that a maximum of 65535 data items can be managed by the DMA in a single transaction, even in peripheral flow control mode.

Note: When configured in memory-to-memory mode, the DMA is always the flow controller and the PFCTRL bit is forced to 0 by hardware.

The circular mode is forbidden in the peripheral flow controller mode.
15.3.18 Summary of the possible DMA configurations

The table below summarizes the different possible DMA configurations. The forbidden configurations are highlighted in gray in the table.

<table>
<thead>
<tr>
<th>DMA transfer mode</th>
<th>Source</th>
<th>Destination</th>
<th>Flow controller</th>
<th>Circular mode</th>
<th>Transfer type</th>
<th>Direct mode</th>
<th>Double-buffer mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peripheral-to-memory</td>
<td>AHB peripheral port</td>
<td>AHB memory port</td>
<td>DMA</td>
<td>Possible</td>
<td>single</td>
<td>Possible</td>
<td>Possible</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>burst</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Peripheral</td>
<td>Forbidden</td>
<td>single</td>
<td>Possible</td>
<td>Forbidden</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>burst</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory-to-peripheral</td>
<td>AHB memory port</td>
<td>AHB peripheral port</td>
<td>DMA</td>
<td>Possible</td>
<td>single</td>
<td>Possible</td>
<td>Possible</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>burst</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Peripheral</td>
<td>Forbidden</td>
<td>single</td>
<td>Possible</td>
<td>Forbidden</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>burst</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory-to-memory</td>
<td>AHB peripheral port</td>
<td>AHB memory port</td>
<td>DMA only</td>
<td>Forbidden</td>
<td>single</td>
<td>Forbidden</td>
<td>Forbidden</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>burst</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 92. Possible DMA configurations

15.3.19 Stream configuration procedure

The following sequence must be followed to configure a DMA stream x (where x is the stream number):

1. If the stream is enabled, disable it by resetting the EN bit in the DMA_SxCR register, then read this bit in order to confirm that there is no ongoing stream operation. Writing this bit to 0 is not immediately effective since it is actually written to 0 once all the current transfers are finished. When the EN bit is read as 0, this means that the stream is ready to be configured. It is therefore necessary to wait for the EN bit to be cleared before starting any stream configuration. All the stream dedicated bits set in the status register (DMA_LISR and DMA_HISR) from the previous data block DMA transfer must be cleared before the stream can be re-enabled.

2. Set the peripheral port register address in the DMA_SxPAR register. The data is moved from/to this address to/from the peripheral port after the peripheral event.

3. Set the memory address in the DMA_SxMA0R register (and in the DMA_SxMA1R register in the case of a double-buffer mode). The data is written to or read from this memory after the peripheral event.

4. Configure the total number of data items to be transferred in the DMA_SxNDTR register. After each peripheral event or each beat of the burst, this value is decremented.

5. Use DMAMUX1 to route a DMA request line to the DMA channel.

6. If the peripheral is intended to be the flow controller and if it supports this feature, set the PFCTRL bit in the DMA_SxCR register.

7. Configure the stream priority using the PL[1:0] bits in the DMA_SxCR register.

8. Configure the FIFO usage (enable or disable, threshold in transmission and reception)
9. Configure the data transfer direction, peripheral and memory incremented/fixed mode, single or burst transactions, peripheral and memory data widths, circular mode, double-buffer mode and interrupts after half and/or full transfer, and/or errors in the DMA_SxCR register.

10. Activate the stream by setting the EN bit in the DMA_SxCR register.

As soon as the stream is enabled, it can serve any DMA request from the peripheral connected to the stream.

Once half the data have been transferred on the AHB destination port, the half-transfer flag (HTIF) is set and an interrupt is generated if the half-transfer interrupt enable bit (HTIE) is set. At the end of the transfer, the transfer complete flag (TCIF) is set and an interrupt is generated if the transfer complete interrupt enable bit (TCIE) is set.

---

**Warning:** To switch off a peripheral connected to a DMA stream request, it is mandatory to, first, switch off the DMA stream to which the peripheral is connected, then to wait for EN bit = 0. Only then can the peripheral be safely disabled.

---

### 15.3.20 Error management

The DMA controller can detect the following errors:

- **Transfer error:** the transfer error interrupt flag (TEIFx) is set when:
  - a bus error occurs during a DMA read or a write access
  - a write access is requested by software on a memory address register in double-buffer mode whereas the stream is enabled and the current target memory is the one impacted by the write into the memory address register (refer to Section 15.3.11: Double-buffer mode)

- **FIFO error:** the FIFO error interrupt flag (FEIFx) is set if:
  - a FIFO underrun condition is detected
  - a FIFO overrun condition is detected (no detection in memory-to-memory mode because requests and transfers are internally managed by the DMA)
  - the stream is enabled while the FIFO threshold level is not compatible with the size of the memory burst (refer to Table 91: FIFO threshold configurations)

- **Direct mode error:** the direct mode error interrupt flag (DMEIFx) can only be set in the peripheral-to-memory mode while operating in direct mode and when the MINC bit in the DMA_SxCR register is cleared. This flag is set when a DMA request occurs while the previous data have not yet been fully transferred into the memory (because the memory bus was not granted). In this case, the flag indicates that 2 data items were transferred successively to the same destination address, which could be an issue if the destination is not able to manage this situation.

  In direct mode, the FIFO error flag can also be set under the following conditions:
  - In the peripheral-to-memory mode, the FIFO can be saturated (overrun) if the memory bus is not granted for several peripheral requests.
  - In the memory-to-peripheral mode, an underrun condition may occur if the memory bus has not been granted before a peripheral request occurs.
If the TEIFx or the FEIFx flag is set due to incompatibility between burst size and FIFO threshold level, the faulty stream is automatically disabled through a hardware clear of its EN bit in the corresponding stream configuration register (DMA_SxCR).

If the DMEIFx or the FEIFx flag is set due to an overrun or underrun condition, the faulty stream is not automatically disabled and it is up to the software to disable or not the stream by resetting the EN bit in the DMA_SxCR register. This is because there is no data loss when this kind of errors occur.

When the stream's error interrupt flag (TEIF, FEIF, DMEIF) in the DMA_LISR or DMA_HISR register is set, an interrupt is generated if the corresponding interrupt enable bit (TEIE, FEIE, DMEIE) in the DMA_SxCR or DMA_SxFCR register is set.

Note: When a FIFO overrun or underrun condition occurs, the data is not lost because the peripheral request is not acknowledged by the stream until the overrun or underrun condition is cleared. If this acknowledge takes too much time, the peripheral itself may detect an overrun or underrun condition of its internal buffer and data might be lost.

### 15.4 DMA interrupts

For each DMA stream, an interrupt can be produced on the following events:

- Half-transfer reached
- Transfer complete
- Transfer error
- FIFO error (overrun, underrun or FIFO level error)
- Direct mode error

Separate interrupt enable control bits are available for flexibility as shown in the table below.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Half-transfer</td>
<td>HTIF</td>
<td>HTIE</td>
</tr>
<tr>
<td>Transfer complete</td>
<td>TCIF</td>
<td>TCIE</td>
</tr>
<tr>
<td>Transfer error</td>
<td>TEIF</td>
<td>TEIE</td>
</tr>
<tr>
<td>FIFO overrun/underrun</td>
<td>FEIF</td>
<td>FEIE</td>
</tr>
<tr>
<td>Direct mode error</td>
<td>DMEIF</td>
<td>DMEIE</td>
</tr>
</tbody>
</table>

Note: Before setting an enable control bit EN = 1, the corresponding event flag must be cleared, otherwise an interrupt is immediately generated.
15.5 DMA registers

The DMA registers have to be accessed by words (32 bits).

15.5.1 DMA low interrupt status register (DMA_LISR)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27, 21, 5</td>
<td>TCIF[3:0]: stream x transfer complete interrupt flag (x = 3 to 0)</td>
</tr>
<tr>
<td>26, 20, 4</td>
<td>HTIF[3:0]: stream x half transfer interrupt flag (x = 3 to 0)</td>
</tr>
<tr>
<td>25, 19, 3</td>
<td>TEIF[3:0]: stream x transfer error interrupt flag (x = 3 to 0)</td>
</tr>
<tr>
<td>24, 18, 2</td>
<td>DMEIF[3:0]: stream x direct mode error interrupt flag (x = 3 to 0)</td>
</tr>
<tr>
<td>23, 17, 1</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22, 16, 0</td>
<td>FEIF[3:0]: stream x FIFO error interrupt flag (x = 3 to 0)</td>
</tr>
</tbody>
</table>

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: no [event] on stream x
1: a [event] occurred on stream x
### 15.5.2 DMA high interrupt status register (DMA_HISR)

Address offset: 0x04  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:28, 15:12</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 27, 21, 11, 5 | **TCIF[7:4]**: stream x transfer complete interrupt flag (x = 7 to 4)  
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.  
0: no transfer complete event on stream x  
1: a transfer complete event occurred on stream x |
| Bit 26, 20, 10, 4 | **HTIF[7:4]**: stream x half transfer interrupt flag (x = 7 to 4)  
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.  
0: no half transfer event on stream x  
1: a half transfer event occurred on stream x |
| Bit 25, 19, 9, 3 | **TEIF[7:4]**: stream x transfer error interrupt flag (x = 7 to 4)  
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.  
0: no transfer error on stream x  
1: a transfer error occurred on stream x |
| Bit 24, 18, 8, 2 | **DMEIF[7:4]**: stream x direct mode error interrupt flag (x = 7 to 4)  
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.  
0: no direct mode error on stream x  
1: a direct mode error occurred on stream x |
| Bit 23, 17, 7, 1 | Reserved, must be kept at reset value. |
| Bit 22, 16, 6, 0 | **FEIF[7:4]**: stream x FIFO error interrupt flag (x = 7 to 4)  
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.  
0: no FIFO error event on stream x  
1: a FIFO error event occurred on stream x |
### 15.5.3 DMA low interrupt flag clear register (DMA_LIFCR)

Address offset: 0x08  
Reset value: 0x0000 0000

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

**Bits 31:28, 15:12**  Reserved, must be kept at reset value.

**Bits 27, 21, 11, 5**  **CTCIF[3:0]:** stream x clear transfer complete interrupt flag (x = 3 to 0)  
Writing 1 to this bit clears the corresponding TCIFx flag in the DMA_LISR register.

**Bits 26, 20, 10, 4**  **CHTIF[3:0]:** stream x clear half transfer interrupt flag (x = 3 to 0)  
Writing 1 to this bit clears the corresponding HTIFx flag in the DMA_LISR register.

**Bits 25, 19, 9, 3**  **CTEIF[3:0]:** stream x clear transfer error interrupt flag (x = 3 to 0)  
Writing 1 to this bit clears the corresponding TEIFx flag in the DMA_LISR register.

**Bits 24, 18, 8, 2**  **CDMEIF[3:0]:** stream x clear direct mode error interrupt flag (x = 3 to 0)  
Writing 1 to this bit clears the corresponding DMEIFx flag in the DMA_LISR register.

**Bits 23, 17, 7, 1**  Reserved, must be kept at reset value.

**Bits 22, 16, 6, 0**  **CFEIF[3:0]:** stream x clear FIFO error interrupt flag (x = 3 to 0)  
Writing 1 to this bit clears the corresponding CFEIFx flag in the DMA_LISR register.

### 15.5.4 DMA high interrupt flag clear register (DMA_HIFCR)

Address offset: 0x0C  
Reset value: 0x0000 0000

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

**Bits 31:28, 15:12**  Reserved, must be kept at reset value.

**Bits 27, 21, 11, 5**  **CTCIF[7:4]:** stream x clear transfer complete interrupt flag (x = 7 to 4)  
Writing 1 to this bit clears the corresponding TCIFx flag in the DMA_HISR register.

**Bits 26, 20, 10, 4**  **CHTIF[7:4]:** stream x clear half transfer interrupt flag (x = 7 to 4)  
Writing 1 to this bit clears the corresponding HTIFx flag in the DMA_HISR register.

**Bits 25, 19, 9, 3**  **CTEIF[7:4]:** stream x clear transfer error interrupt flag (x = 7 to 4)  
Writing 1 to this bit clears the corresponding TEIFx flag in the DMA_HISR register.
15.5.5 DMA stream x configuration register (DMA_SxCR)

This register is used to configure the concerned stream.

Address offset: 0x10 + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBURST[1:0]</td>
<td>PBURST[1:0]</td>
<td>CT</td>
<td>DBM</td>
<td>PL[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 24, 18, 8, 2 \textbf{CDMEIF}[7:4]: stream x clear direct mode error interrupt flag (x = 7 to 4)

Writing 1 to this bit clears the corresponding DMEIFx flag in the DMA_HISR register.

Bits 23, 17, 7, 1 \textbf{Reserved}, must be kept at reset value.

Bits 22, 16, 6, 0 \textbf{CFEIF}[7:4]: stream x clear FIFO error interrupt flag (x = 7 to 4)

Writing 1 to this bit clears the corresponding CFEIFx flag in the DMA_HISR register.

Bits 31:25 \textbf{Reserved}, must be kept at reset value.

Bits 24:23 \textbf{MBURST}[1:0]: memory burst transfer configuration

These bits are set and cleared by software.

00: single transfer
01: INCR4 (incremental burst of 4 beats)
10: INCR8 (incremental burst of 8 beats)
11: INCR16 (incremental burst of 16 beats)

These bits are protected and can be written only if EN is ‘0’.

In direct mode, these bits are forced to 0x0 by hardware as soon as bit EN= ‘1’.

Bits 22:21 \textbf{PBURST}[1:0]: peripheral burst transfer configuration

These bits are set and cleared by software.

00: single transfer
01: INCR4 (incremental burst of 4 beats)
10: INCR8 (incremental burst of 8 beats)
11: INCR16 (incremental burst of 16 beats)

These bits are protected and can be written only if EN is ‘0’.

In direct mode, these bits are forced to 0x0 by hardware.

Bit 20 \textbf{Reserved}, must be kept at reset value.

Bit 19 \textbf{CT}: current target (only in double-buffer mode)

This bit is set and cleared by hardware. It can also be written by software.

0: current target memory is Memory 0 (addressed by the DMA_SxM0AR pointer)
1: current target memory is Memory 1 (addressed by the DMA_SxM1AR pointer)

This bit can be written only if EN is ‘0’ to indicate the target memory area of the first transfer.

Once the stream is enabled, this bit operates as a status flag indicating which memory area is the current target.
Bit 18  **DBM**: double-buffer mode

This bit is set and cleared by software.

0: no buffer switching at the end of transfer
1: memory target switched at the end of the DMA transfer

This bit is protected and can be written only if EN is ‘0’.

Bits 17:16  **PL[1:0]**: priority level

These bits are set and cleared by software.

00: low
01: medium
10: high
11: very high

These bits are protected and can be written only if EN is ‘0’.

Bit 15  **PINCOS**: peripheral increment offset size

This bit is set and cleared by software

0: The offset size for the peripheral address calculation is linked to the PSIZE
1: The offset size for the peripheral address calculation is fixed to 4 (32-bit alignment).

This bit has no meaning if bit PINC = ‘0’.

This bit is protected and can be written only if EN = ‘0’.

This bit is forced low by hardware when the stream is enabled (bit EN = ‘1’) if the direct mode is selected or if PBURST are different from “00”.

Bits 14:13  **MSIZE[1:0]**: memory data size

These bits are set and cleared by software.

00: byte (8-bit)
01: half-word (16-bit)
10: word (32-bit)
11: reserved

These bits are protected and can be written only if EN is ‘0’.

In direct mode, MSIZE is forced by hardware to the same value as PSIZE as soon as bit EN = ‘1’.

Bits 12:11  **PSIZE[1:0]**: peripheral data size

These bits are set and cleared by software.

00: byte (8-bit)
01: half-word (16-bit)
10: word (32-bit)
11: reserved

These bits are protected and can be written only if EN is ‘0’.

Bit 10  **M INC**: memory increment mode

This bit is set and cleared by software.

0: memory address pointer is fixed
1: memory address pointer is incremented after each data transfer (increment is done according to MSIZE)

This bit is protected and can be written only if EN is ‘0’.

Bit 9  **PINC**: peripheral increment mode

This bit is set and cleared by software.

0: peripheral address pointer is fixed
1: peripheral address pointer is incremented after each data transfer (increment is done according to PSIZE)

This bit is protected and can be written only if EN is ‘0’.
Bit 8  **CIRC**: circular mode
This bit is set and cleared by software and can be cleared by hardware.
0: circular mode disabled
1: circular mode enabled
When the peripheral is the flow controller (bit PFCTRL = 1) and the stream is enabled (bit EN = 1), then this bit is automatically forced by hardware to 0.
It is automatically forced by hardware to 1 if the DBM bit is set, as soon as the stream is enabled (bit EN =”1”).

Bits 7:6  **DIR[1:0]**: data transfer direction
These bits are set and cleared by software.
00: peripheral-to-memory
01: memory-to-peripheral
10: memory-to-memory
11: reserved
These bits are protected and can be written only if EN is ‘0’.

Bit 5  **PFCTRL**: peripheral flow controller
This bit is set and cleared by software.
0: DMA is the flow controller
1: The peripheral is the flow controller
This bit is protected and can be written only if EN is ‘0’.
When the memory-to-memory mode is selected (bits DIR[1:0]=10), then this bit is automatically forced to 0 by hardware.

Bit 4  **TCIE**: transfer complete interrupt enable
This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 3  **HTIE**: half transfer interrupt enable
This bit is set and cleared by software.
0: HT interrupt disabled
1: HT interrupt enabled

Bit 2  **TEIE**: transfer error interrupt enable
This bit is set and cleared by software.
0: TE interrupt disabled
1: TE interrupt enabled

Bit 1  **DMEIE**: direct mode error interrupt enable
This bit is set and cleared by software.
0: DME interrupt disabled
1: DME interrupt enabled
Bit 0  **EN**: stream enable / flag stream ready when read low
This bit is set and cleared by software.
0: stream disabled
1: stream enabled
This bit may be cleared by hardware:
– on a DMA end of transfer (stream ready to be configured)
– if a transfer error occurs on the AHB master buses
– when the FIFO threshold on memory AHB port is not compatible with the size of the burst
When this bit is read as 0, the software is allowed to program the configuration and FIFO bits registers. It is forbidden to write these registers when the EN bit is read as 1.

**Note**: Before setting EN bit to ‘1’ to start a new transfer, the event flags corresponding to the stream in DMA_LISR or DMA_HISR register must be cleared.

### 15.5.6 DMA stream x number of data register (DMA_SxNDTR)

Address offset: 0x14 + 0x18 * x, (x = 0 to 7)
Reset value: 0x0000 0000

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

**NDT[15:0]**

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **NDT[15:0]**: number of data items to transfer (0 up to 65535)
This register can be written only when the stream is disabled. When the stream is enabled, this register is read-only, indicating the remaining data items to be transmitted. This register decrements after each DMA transfer.
Once the transfer is completed, this register can either stay at zero (when the stream is in normal mode) or be reloaded automatically with the previously programmed value in the following cases:
– when the stream is configured in circular mode.
– when the stream is enabled again by setting EN bit to ‘1’.
If the value of this register is zero, no transaction can be served even if the stream is enabled.
15.5.7 DMA stream x peripheral address register (DMA_SxPAR)

Address offset: 0x18 + 0x18 * x, (x = 0 to 7)
Reset value: 0x0000 0000

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

Bits 31:0 PAR[31:0]: peripheral address
Base address of the peripheral data register from/to which the data is read/written.
These bits are write-protected and can be written only when bit EN = '0' in the DMA_SxCR register.

15.5.8 DMA stream x memory 0 address register (DMA_SxM0AR)

Address offset: 0x1C + 0x18 * x, (x = 0 to 7)
Reset value: 0x0000 0000

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

Bits 31:0 M0A[31:0]: memory 0 address
Base address of memory area 0 from/to which the data is read/written.
These bits are write-protected. They can be written only if:
- the stream is disabled (bit EN = '0' in the DMA_SxCR register) or
- the stream is enabled (EN='1' in DMA_SxCR register) and bit CT = '1' in the DMA_SxCR register (in double-buffer mode).
15.5.9 DMA stream x memory 1 address register (DMA_SxM1AR)

Address offset: 0x20 + 0x18 * x, (x = 0 to 7)
Reset value: 0x0000 0000

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

Bits 31:0 M1A[31:0]: memory 1 address (used in case of double-buffer mode)
Base address of memory area 1 from/to which the data is read/written.
This register is used only for the double-buffer mode.
These bits are write-protected. They can be written only if:
– the stream is disabled (bit EN= '0' in the DMA_SxCR register) or
– the stream is enabled (EN='1' in DMA_SxCR register) and bit CT = '0' in the DMA_SxCR register.

15.5.10 DMA stream x FIFO control register (DMA_SxFCR)

Address offset: 0x24 + 0x18 * x, (x = 0 to 7)
Reset value: 0x0000 0021

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

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

Bit 7 FEIE: FIFO error interrupt enable
This bit is set and cleared by software.
0: FE interrupt disabled
1: FE interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bits 5:3 FS[2:0]: FIFO status
These bits are read-only.
000: 0 < fifo_level < 1/4
001: 1/4 ≤ fifo_level < 1/2
010: 1/2 ≤ fifo_level < 3/4
011: 3/4 ≤ fifo_level < full
100: FIFO is empty
101: FIFO is full
others: no meaning
These bits are not relevant in the direct mode (DMDIS bit is zero).
Bit 2  **DMDIS**: direct mode disable
   This bit is set and cleared by software. It can be set by hardware.
   0: direct mode enabled
   1: direct mode disabled
   This bit is protected and can be written only if EN is ‘0’.
   This bit is set by hardware if the memory-to-memory mode is selected (DIR bit in DMA_SxCR are “10”) and the EN bit in the DMA_SxCR register is ‘1’ because the direct mode is not allowed in the memory-to-memory configuration.

Bits 1:0  **FTH[1:0]**: FIFO threshold selection
   These bits are set and cleared by software.
   00: 1/4 full FIFO
   01: 1/2 full FIFO
   10: 3/4 full FIFO
   11: full FIFO
   These bits are not used in the direct mode when the DMIS value is zero.
   These bits are protected and can be written only if EN is ‘0’.
## 15.5.11 DMA register map

The table below summarizes the DMA registers.

### Table 94. DMA register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x0000 | DMA_LISR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 0x0004        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 0x0008        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 0x000C        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 0x0010        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 0x0014        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 0x0018        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 0x001C        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 0x0020        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 0x0024        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 0x0028        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 0x002C        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Table 94. DMA register map and reset values (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0030</td>
<td>DMA_S1PAR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0034</td>
<td>DMA_S1M0AR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0038</td>
<td>DMA_S1M1AR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x003C</td>
<td>DMA_S1FCR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0040</td>
<td>DMA_S2CR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0044</td>
<td>DMA_S2NDTR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0048</td>
<td>DMA_S2PAR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x004C</td>
<td>DMA_S2M0AR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0050</td>
<td>DMA_S2M1AR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0054</td>
<td>DMA_S2FCR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0058</td>
<td>DMA_S3CR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x005C</td>
<td>DMA_S3NDTR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0060</td>
<td>DMA_S3PAR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0064</td>
<td>DMA_S3M0AR</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
<td>0x0000</td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>-------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0068</td>
<td>DMA_S3M1AR</td>
<td>M1A[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x006C</td>
<td>DMA_S3FCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0070</td>
<td>DMA_S4CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0074</td>
<td>DMA_S4NDTR</td>
<td>NDT[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0078</td>
<td>DMA_S4PAR</td>
<td>PA[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x007C</td>
<td>DMA_S4M0AR</td>
<td>M0A[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0080</td>
<td>DMA_S4M1AR</td>
<td>M1A[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0084</td>
<td>DMA_S4FCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0088</td>
<td>DMA_S5CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008C</td>
<td>DMA_S5NDTR</td>
<td>NDT[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0090</td>
<td>DMA_S5PAR</td>
<td>PA[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0094</td>
<td>DMA_S5M0AR</td>
<td>M0A[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0098</td>
<td>DMA_S5M1AR</td>
<td>M1A[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x009C</td>
<td>DMA_S5FCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Refer to Section 2.3 on page 128 for the register boundary addresses.
16 Basic direct memory access controller (BDMA)

16.1 Introduction

The basic direct memory access (BDMA) controller is a bus master and system peripheral. The BDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories, upon the control of an off-loaded CPU.

The BDMA controller features a single AHB master architecture.

There are two instances of BDMA, BDMA1 (dedicated to the 8-filter DFSDM) and BDMA2, with 8 channels each.

Each channel is dedicated to managing memory access requests from one or more peripherals. Each BDMA includes an arbiter for handling the priority between DMA requests.

16.2 BDMA main features

- Single AHB master
- Peripheral-to-memory, memory-to-peripheral, memory-to-memory and peripheral-to-peripheral data transfers
- Access to AHB SRAM1/SRAM2 and DFSDM peripheral (BDMA1)
- Access to SmartRun domain SRAM and AHB/APB peripherals (BDMA2)
- All BDMA channels independently configurable:
  - Each channel is associated either with a DMA request signal coming from a peripheral, or with a software trigger in memory-to-memory transfers. This configuration is done by software.
  - Priority between the requests is programmable by software (4 levels per channel: very high, high, medium, low) and by hardware in case of equality (such as request to channel 1 has priority over request to channel 2).
  - Transfer size of source and destination are independent (byte, half-word, word), emulating packing and unpacking. Source and destination addresses must be aligned on the data size.
  - Support of transfers from/to peripherals to/from memory with circular buffer management
  - Programmable number of data to be transferred: 0 to $2^{16} - 1$
- Generation of an interrupt request per channel. Each interrupt request is caused from any of the three DMA events: transfer complete, half transfer, or transfer error.
16.3 BDMA implementation

16.3.1 BDMA1 and BDMA2

BDMA1 and BDMA2 are implemented with the hardware configuration parameters shown in the table below.

Table 95. BDMA1 and BDMA2 implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>BDMA1</th>
<th>BDMA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels (double-buffer)</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

16.3.2 BDMA request mapping

The BDMA2 controller is connected to DMA requests from the AHB/APB peripherals through the DMAMUX2 peripheral.

For the mapping of the different requests, refer to the *Section 17.3: DMAMUX implementation*.

A channel request \( x \) of the BDMA1 is connected to the DFSDM request of the same index \( x \).
16.4  BDMA functional description

16.4.1  BDMA block diagram

The BDMA block diagram is shown in the figure below.
The BDMA controller performs direct memory transfer by sharing the AHB system bus with other system masters. The bus matrix implements round-robin scheduling. DMA requests may stop the CPU access to the system bus for a number of bus cycles, when CPU and BDMA target the same destination (memory or peripheral).
The BDMA2 controller is connected to DMA requests from the AHB/APB peripherals through the DMAMUX2 peripheral.

A channel request \( x \) of the BDMA1 is connected to the DFSDM request of the same index \( x \).

According to its configuration through the AHB slave interface, the BDMA controller arbitrates between the DMA channels and their associated received requests. The BDMA controller also schedules the DMA data transfers over the single AHB port master.

The BDMA2 controller generates an interrupt per channel to the interrupt controller and the BDMA1 dedicated to DFSDM generates a single ORed interrupt.

### 16.4.2 BDMA pins and internal signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bdma_req[x]</td>
<td>Input</td>
<td>BDMA channel ( x ) request</td>
</tr>
<tr>
<td>bdma_ack[x]</td>
<td>Output</td>
<td>BDMA channel ( x ) acknowledge</td>
</tr>
<tr>
<td>bdma_it[x]</td>
<td>Output</td>
<td>BDMA channel ( x ) interrupt</td>
</tr>
</tbody>
</table>

### 16.4.3 BDMA transfers

The software configures the BDMA controller at channel level, in order to perform a block transfer, composed of a sequence of AHB bus transfers.

A BDMA block transfer may be requested from a peripheral, or triggered by the software in case of memory-to-memory transfer.

After an event, the following steps of a single BDMA transfer occur:

1. The peripheral sends a single DMA request signal to the BDMA controller.
2. The BDMA controller serves the request, depending on the priority of the channel associated to this peripheral request.
3. As soon as the BDMA controller grants the peripheral, an acknowledge is sent to the peripheral by the BDMA controller.
4. The peripheral releases its request as soon as it gets the acknowledge from the BDMA controller.
5. Once the request is deasserted by the peripheral, the BDMA controller releases the acknowledge.

The peripheral may order a further single request and initiate another single BDMA transfer. The request/acknowledge protocol is used when a peripheral is either the source or the destination of the transfer. For example, in case of memory-to-peripheral transfer, the peripheral initiates the transfer by driving its single request signal to the BDMA controller. The BDMA controller reads then a single data in the memory and writes this data to the peripheral.

For a given channel \( x \), a BDMA block transfer consists of a repeated sequence of:

- a single BDMA transfer, encapsulating two AHB transfers of a single data, over the BDMA AHB bus master:
  - a single data read (byte, half-word or word) from the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory...
address register.
The start address used for the first single transfer is the base address of the peripheral or memory, and is programmed in the BDMA_CPARx or BDMA_CM0/1ARx register.

- a single data write (byte, half-word or word) to the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.

The start address used for the first transfer is the base address of the peripheral or memory, and is programmed in the BDMA_CPARx or BDMA_CM0/1ARx register.

- post-decrementing of the programmed BDMA_CNDTRx register

This register contains the remaining number of data items to transfer (number of AHB ‘read followed by write’ transfers).

This sequence is repeated until BDMA_CNDTRx is null.

Note: The AHB master bus source/destination address must be aligned with the programmed size of the transferred single data to the source/destination.

16.4.4 BDMA arbitration

The BDMA arbiter manages the priority between the different channels.

When an active channel x is granted by the arbiter (hardware requested or software triggered), a single BDMA transfer is issued (such as an AHB ‘read followed by write’ transfer of a single data). Then, the arbiter considers again the set of active channels and selects the one with the highest priority.

The priorities are managed in two stages:

- software: priority of each channel is configured in the BDMA_CCRx register, to one of the four different levels:
  - very high
  - high
  - medium
  - low

- hardware: if two requests have the same software priority level, the channel with the lowest index gets priority. For example, channel 2 gets priority over channel 4.

When a channel x is programmed for a block transfer in memory-to-memory mode, re arbitration is considered between each single BDMA transfer of this channel x. Whenever there is another concurrent active requested channel, the BDMA arbiter automatically alternates and grants the other highest-priority requested channel, which may be of lower priority than the memory-to-memory channel.

16.4.5 BDMA channels

Each channel may handle a BDMA transfer between a peripheral register located at a fixed address, and a memory address. The amount of data items to transfer is programmable. The register that contains the amount of data items to transfer is decremented after each transfer.

A DMA channel is programmed at block transfer level.
Programmable data sizes

The transfer sizes of a single data (byte, half-word, or word) to the peripheral and memory are programmable through, respectively, the PSIZE[1:0] and MSIZE[1:0] fields of the BDMA_CCRx register.

Pointer incrementation

The peripheral and memory pointers may be automatically incremented after each transfer, depending on the PINC and MINC bits of the BDMA_CCRx register.

If the **incremented mode** is enabled (PINC or MINC set to 1), the address of the next transfer is the address of the previous one incremented by 1, 2 or 4, depending on the data size defined in PSIZE[1:0] or MSIZE[1:0]. The first transfer address is the one programmed in the BDMA_CPARx or BDMA_CM0/1ARx register. During transfers, these registers keep the initially programmed value. The current transfer addresses (in the current internal peripheral/memory address register) are not accessible by software.

If the channel x is configured in **non-circular mode**, no DMA request is served after the last data transfer (once the number of single data to transfer reaches zero). The DMA channel must be disabled in order to reload a new number of data items into the BDMA_CNDTRx register.

**Note:** If the channel x is disabled, the BDMA registers are not reset. The DMA channel registers (BDMA_CCRx, BDMA_CPARx and BDMA_CM0ARx) retain the initial values programmed during the channel configuration phase.

In **circular mode**, after the last data transfer, the BDMA_CNDTRx register is automatically reloaded with the initially programmed value. The current internal address registers are reloaded with the base address values from the BDMA_CPARx and BDMA_CM0/1ARx registers.

Channel configuration procedure

The following sequence is needed to configure a DMA channel x:

1. Set the peripheral register address in the BDMA_CPARx register. The data is moved from/to this address to/from the memory after the peripheral event, or after the channel is enabled in memory-to-memory mode.
2. Set the memory address in the BDMA_CM0ARx register. The data is written to/read from the memory after the peripheral event or after the channel is enabled in memory-to-memory mode.
3. Configure the total number of data to transfer in the BDMA_CNDTRx register. After each data transfer, this value is decremented.
4. Configure the parameters listed below in the BDMA_CCRx register:
   - the channel priority
   - the data transfer direction
   - the circular mode
   - the double-buffer mode
   - the peripheral and memory incremented mode
   - the peripheral and memory data size
   - the interrupt enable at half and/or full transfer and/or transfer error
5. Activate the channel by setting the EN bit in the BDMA_CCRx register.
A channel, as soon as enabled, may serve any BDMA request from the peripheral connected to this channel, or may start a memory-to-memory block transfer.

**Note:** The two last steps of the channel configuration procedure may be merged into a single access to the BDMA_CCRx register, to configure and enable the channel.

### Channel state and disabling a channel

A channel x in active state is an enabled channel (read BDMA_CCRx.EN = 1). An active channel x is a channel that must have been enabled by the software (BDMA_CCRx.EN set to 1) and afterwards with no occurred transfer error (BDMA_ISR.TEIFx = 0). In case there is a transfer error, the channel is automatically disabled by hardware (BDMA_CCRx.EN = 0).

The three following use cases may happen:

- **Suspend and resume a channel**
  This corresponds to the two following actions:
  - An active channel is disabled by software (writing BDMA_CCRx.EN = 0 whereas BDMA_CCRx.EN = 1).
  - The software enables the channel again (BDMA_CCRx.EN set to 1) without reconfiguring the other channel registers (such as BDMA_CNDTRx, BDMA_CPARx and BDMA_CM0/1ARx).
  This case is not supported by the BDMA hardware, that does not guarantee that the remaining data transfers are performed correctly.

- **Stop and abort a channel**
  If the application does not need any more the channel, this active channel can be disabled by software. The channel is stopped and aborted but the BDMA_CNDTRx register content may not correctly reflect the remaining data transfers versus the aborted source and destination buffer/register.

- **Abort and restart a channel**
  This corresponds to the software sequence: disable an active channel, then reconfigure the channel and enable it again.
  This is supported by the hardware if the following conditions are met:
  - The application guarantees that, when the software is disabling the channel, a DMA data transfer is not occurring at the same time over its master port. For example, the application can first disable the peripheral in DMA mode, in order to ensure that there is no pending hardware DMA request from this peripheral.
  - The software must operate separated write accesses to the same BDMA_CCRx register: First disable the channel. Second reconfigure the channel for a next block transfer including the BDMA_CCRx if a configuration change is needed. There are read-only BDMA_CCRx register fields when BDMA_CCRx.EN=1. Finally enable again the channel.

When a channel transfer error occurs, the EN bit of the BDMA_CCRx register is cleared by hardware. This EN bit can not be set again by software to re-activate the channel x, until the TEIFx bit of the BDMA_CxISR register is set.

### Circular mode (in memory-to-peripheral/peripheral-to-memory transfers)

The circular mode is available to handle circular buffers and continuous data flows (such as ADC scan mode). This feature is enabled using the CIRC bit in the BDMA_CCRx register.
The circular mode must not be used in memory-to-memory mode. Before enabling a channel in circular mode (CIRC = 1), the software must clear the MEM2MEM bit of the BDMA_CCRx register. When the circular mode is activated, the amount of data to transfer is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

In order to stop a circular transfer, the software needs to stop the peripheral from generating DMA requests (such as quit the ADC scan mode), before disabling the DMA channel. The software must explicitly program the BDMA_CNDTRx value before starting/enabling a transfer, and after having stopped a circular transfer.

Memory-to-memory mode

The BDMA channels may operate without being triggered by a request from a peripheral. This mode is called memory-to-memory mode, and is initiated by software.

If the MEM2MEM bit in the BDMA_CCRx register is set, the channel, if enabled, initiates transfers. The transfer stops once the BDMA_CNDTRx register reaches zero.

Note: The memory-to-memory mode must not be used in circular mode. Before enabling a channel in memory-to-memory mode (MEM2MEM = 1), the software must clear the CIRC bit of the BDMA_CCRx register.

Double-buffer mode (in memory-to-peripheral and peripheral-to-memory transfers)

The BDMA channels can operate in double-buffer mode.

The difference compared to a regular operation is that the BDMA controller toggles between two memory address pointers at the end of each BDMA transfer, thus accessing two memory areas in an alternate way. This allows the software to access one of the two memory areas while the BDMA controller accesses the other one. The double-buffer mode transfer operates in both directions, so the target memory can be either the source or the destination.

The double-buffer mode is configured by setting both the DBM and CIRC bits of the BDMA_CCRx register.

Note: The double-buffer mode must not be used in memory-to-memory mode. Before enabling a channel in double-buffer mode (DBM = 1), the software has to configure appropriately the MEM2MEM bit (MEM2MEM = 0).

The steps described below allow the configuration of a BDMA channel x in double-buffer mode:

- Set the DBM and CIRC bits and clear the MEM2MEM bit of the BDMA_CCRx register. The circular mode is then activated for the swap mechanism to occur.
- Configure the second memory address register BDMA_CM1ARx.
- Continue with the regular channel configuration procedure, and lastly, activate the channel by setting the EN bit of the BDMA_CCRx register. The first BDMA transfer target memory of the corresponding BDMA channel x, is given by the CT bit of the BDMA_CCRx register.

Note: Independently from the value of DBM bit of the BDMA_CCRx register, if CT = 1, the memory address pointer for the BDMA transfer is defined by BDMA_CM1ARx, and not by BDMA_CM0ARx.
Peripheral-to-peripheral mode

Any BDMA channel can operate in peripheral-to-peripheral mode:

- when the hardware request from a peripheral is selected to trigger the BDMA channel
  This peripheral is the BDMA initiator and paces the data transfer from/to this peripheral
  to/from a register belonging to another memory-mapped peripheral (this one being not
  configured in DMA mode).
- when no peripheral request is selected and connected to the BDMA channel
  The software configures a register-to-register transfer by setting the MEM2MEM bit of
  the BDMA_CCRx register.

Programming transfer direction, assigning source/destination

The value of the DIR bit of the BDMA_CCRx register sets the direction of the transfer, and
consequently, it identifies the source and the destination, regardless the source/destination
type (peripheral or memory):

- **DIR = 1** defines typically a memory-to-peripheral transfer. More generally, if DIR = 1:
  - The **source** attributes are defined by the BDMA_MARx register, the MSIZE[1:0]
    field and MINC bit of the BDMA_CCRx register.
    Regardless of their usual naming, these ‘memory’ register, field and bit are used to
    define the source peripheral in peripheral-to-peripheral mode.
  - The **destination** attributes are defined by the BDMA_PARx register, the
    PSIZE[1:0] field and PINC bit of the BDMA_CCRx register.
    Regardless of their usual naming, these ‘peripheral’ register, field and bit are used
    to define the destination memory in memory-to-memory mode.

- **DIR = 0** defines typically a peripheral-to-memory transfer. More generally, if DIR = 0:
  - The **source** attributes are defined by the BDMA_PARx register, the PSIZE[1:0]
    field and PINC bit of the BDMA_CCRx register.
    Regardless of their usual naming, these ‘peripheral’ register, field and bit are used
    to define the source memory in memory-to-memory mode
  - The **destination** attributes are defined by the BDMA_MARx register, the
    MSIZE[1:0] field and MINC bit of the BDMA_CCRx register.
    Regardless of their usual naming, these ‘memory’ register, field and bit are used to
    define the destination peripheral in peripheral-to-peripheral mode.
16.4.6 BDMA data width, alignment and endianness

When PSIZE[1:0] and MSIZE[1:0] are not equal, the BDMA controller performs some data alignments as described in the table below.

<table>
<thead>
<tr>
<th>Source port width (MSIZE if DIR = 1, else PSIZE)</th>
<th>Destination port width (MSIZE if DIR = 1, else PSIZE)</th>
<th>Number of data items to transfer (NDT)</th>
<th>Source content: address / data (BDMA_CM0/1ARx if DIR = 1, else BDMA_CPARx)</th>
<th>DMA transfers</th>
<th>Destination content: address / data (BDMA_CPARx if DIR = 1, else BDMA_CM0/1ARx)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>@0x0 / B0</td>
<td>1: read B0[7:0] @0x0 then write B0[7:0] @0x0</td>
<td>@0x0 / B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x1 / B1</td>
<td>2: read B1[7:0] @0x1 then write B1[7:0] @0x1</td>
<td>@0x1 / B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x2 / B2</td>
<td>3: read B2[7:0] @0x2 then write B2[7:0] @0x2</td>
<td>@0x2 / B2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x3 / B3</td>
<td>4: read B3[7:0] @0x3 then write B3[7:0] @0x3</td>
<td>@0x3 / B3</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>4</td>
<td>@0x0 / B0</td>
<td>1: read B0[7:0] @0x0 then write 00B0[15:0] @0x0</td>
<td>@0x0 / 00B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x1 / B1</td>
<td>2: read B1[7:0] @0x1 then write 00B1[15:0] @0x2</td>
<td>@0x2 / 00B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x2 / B2</td>
<td>3: read B2[7:0] @0x2 then write 00B2[15:0] @0x4</td>
<td>@0x4 / 00B2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x3 / B3</td>
<td>4: read B3[7:0] @0x3 then write 00B3[15:0] @0x6</td>
<td>@0x6 / 00B3</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>4</td>
<td>@0x0 / B0</td>
<td>1: read B0[7:0] @0x0 then write 000000B0[31:0] @0x0</td>
<td>@0x0 / 000000B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x1 / B1</td>
<td>2: read B1[7:0] @0x1 then write 000000B1[31:0] @0x4</td>
<td>@0x4 / 000000B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x2 / B2</td>
<td>3: read B2[7:0] @0x2 then write 000000B2[31:0] @0x8</td>
<td>@0x8 / 000000B2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x3 / B3</td>
<td>4: read B3[7:0] @0x3 then write 000000B3[31:0] @0xC</td>
<td>@0xC / 000000B3</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>4</td>
<td>@0x0 / B1B0</td>
<td>1: read B1B0[15:0] @0x0 then write B0[7:0] @0x0</td>
<td>@0x0 / B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x1 / B1B2</td>
<td>2: read B3B2[15:0] @0x2 then write B2[7:0] @0x1</td>
<td>@0x1 / B2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x2 / B3B4</td>
<td>3: read B5B4[15:0] @0x4 then write B4[7:0] @0x2</td>
<td>@0x2 / B4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x3 / B7B6</td>
<td>4: read B7B6[15:0] @0x6 then write B6[7:0] @0x3</td>
<td>@0x3 / B6</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>4</td>
<td>@0x0 / BIB0</td>
<td>1: read B1B0[15:0] @0x0 then write 0000B1B0[31:0] @0x0</td>
<td>@0x0 / 0000B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x1 / B3B2</td>
<td>2: read B3B2[15:0] @0x2 then write 0000B3B2[31:0] @0x4</td>
<td>@0x4 / 0000B3B2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x2 / B5B4</td>
<td>3: read B5B4[15:0] @0x4 then write 0000B5B4[31:0] @0x8</td>
<td>@0x8 / 0000B5B4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x3 / B7B6</td>
<td>4: read B7B6[15:0] @0x6 then write 0000B7B6[31:0] @0xC</td>
<td>@0xC / 0000B7B6</td>
</tr>
<tr>
<td>16</td>
<td>32</td>
<td>4</td>
<td>@0x0 / BIB0</td>
<td>1: read B1B0[15:0] @0x0 then write 0000B1B0[31:0] @0x0</td>
<td>@0x0 / 0000B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x1 / B3B2</td>
<td>2: read B3B2[15:0] @0x2 then write 0000B3B2[31:0] @0x4</td>
<td>@0x4 / 0000B3B2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x2 / B5B4</td>
<td>3: read B5B4[15:0] @0x4 then write 0000B5B4[31:0] @0x8</td>
<td>@0x8 / 0000B5B4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x3 / B7B6</td>
<td>4: read B7B6[15:0] @0x6 then write 0000B7B6[31:0] @0xC</td>
<td>@0xC / 0000B7B6</td>
</tr>
<tr>
<td>32</td>
<td>8</td>
<td>4</td>
<td>@0x0 / B3B2B1B0</td>
<td>1: read B3B2B1B0[31:0] @0x0 then write B0[7:0] @0x0</td>
<td>@0x0 / B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x1 / B3B2B1B0</td>
<td>2: read B7B6B5B4[31:0] @0x4 then write B4[7:0] @0x1</td>
<td>@0x1 / B4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x2 / B3B2B1B0</td>
<td>3: read B8B9B8B9B8[31:0] @0x8 then write B7B6B5B4[31:0] @0x2</td>
<td>@0x2 / B8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x3 / B3B2B1B0</td>
<td>4: read BFBEBD[31:0] @0xC then write BC[7:0] @0x3</td>
<td>@0x3 / BC</td>
</tr>
<tr>
<td>32</td>
<td>16</td>
<td>4</td>
<td>@0x0 / B3B2B1B0</td>
<td>1: read B3B2B1B0[31:0] @0x0 then write B1B0[15:0] @0x0</td>
<td>@0x0 / B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x1 / B7B6B5B4</td>
<td>2: read B7B6B5B4[31:0] @0x4 then write B5B4[15:0] @0x2</td>
<td>@0x2 / B5B4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x2 / B3B2B1B0</td>
<td>3: read B8B9B8B9B8[31:0] @0x8 then write B8B9B8[15:0] @0x4</td>
<td>@0x4 / B8B9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x3 / B3B2B1B0</td>
<td>4: read BFBEBD[31:0] @0xC then write BD[15:0] @0x6</td>
<td>@0x6 / BD[BC]</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>4</td>
<td>@0x0 / B3B2B1B0</td>
<td>1: read B3B2B1B0[31:0] @0x0 then write B3B2B1B0[31:0] @0x0</td>
<td>@0x0 / B3B2B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x1 / B7B6B5B4</td>
<td>2: read B7B6B5B4[31:0] @0x4 then write B7B6B5B4[31:0] @0x4</td>
<td>@0x4 / B7B6B5B4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x2 / B3B2B1B0</td>
<td>3: read B8B9B8B9B8[31:0] @0x8 then write B8B9B8B9B8[31:0] @0x8</td>
<td>@0xB / B8B9B8B9B8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@0x3 / B3B2B1B0</td>
<td>4: read BFBEBD[31:0] @0xC then write BFBEBD[31:0] @0xC</td>
<td>@0xC / BFBEBD[31:0]</td>
</tr>
</tbody>
</table>
Addressing AHB peripherals not supporting byte/half-word write transfers

When the BDMA controller initiates an AHB byte or half-word write transfer, the data are duplicated on the unused lanes of the AHB master 32-bit data bus (HWDATA[31:0]).

When the AHB slave peripheral does not support byte or half-word write transfers and does not generate any error, the BDMA controller writes the 32 HWDATA bits as shown in the two examples below:

- To write the half-word 0xABCD, the BDMA controller sets the HWDATA bus to 0xABCDABCD with a half-word data size (HSIZE = HalfWord in AHB master bus).
- To write the byte 0xAB, the BDMA controller sets the HWDATA bus to 0xABABABAB with a byte data size (HSIZE = Byte in the AHB master bus).

Assuming the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take into account the HSIZE data, any AHB byte or half-word transfer is changed into a 32-bit APB transfer as described below:

- An AHB byte write transfer of 0xB0 to one of the 0x0, 0x1, 0x2 or 0x3 addresses, is converted to an APB word write transfer of 0xB0B0B0B0 to the 0x0 address.
- An AHB half-word write transfer of 0xB1B0 to the 0x0 or 0x2 addresses, is converted to an APB word write transfer of 0xB1B0B1B0 to the 0x0 address.

16.4.7 BDMA error management

A BDMA transfer error is generated when reading from or writing to a reserved address space. When a BDMA transfer error occurs during a BDMA read or write access, the faulty channel $x$ is automatically disabled through a hardware clear of its EN bit in the corresponding BDMA_CCRx register.

The TEIFx bit of the BDMA_ISR register is set. An interrupt is then generated if the TEIE bit of the BDMA_CCRx register is set.

The EN bit of the BDMA_CCRx register can not be set again by software (channel $x$ re-activated) until the TEIFx bit of the BDMA_ISR register is cleared (by setting the CTEIFx bit of the BDMA_IFCR register).

When the software is notified with a transfer error over a channel which involves a peripheral, the software has first to stop this peripheral in DMA mode, in order to disable any pending or future DMA request. Then software may normally reconfigure both BDMA and the peripheral in DMA mode for a new transfer.
16.5  BDMA interrupts

An interrupt can be generated on a half transfer, transfer complete or transfer error for each DMA channel x. Separate interrupt enable bits are available for flexibility.

Table 98. BDMA interrupt requests

<table>
<thead>
<tr>
<th>Interrupt request</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Interrupt enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel x interrupt</td>
<td>Half transfer on channel x</td>
<td>HTIFx</td>
<td>HTIEx</td>
</tr>
<tr>
<td></td>
<td>Transfer complete on channel x</td>
<td>TCIFx</td>
<td>TCIEx</td>
</tr>
<tr>
<td></td>
<td>Transfer error on channel x</td>
<td>TEIFx</td>
<td>TEIEx</td>
</tr>
<tr>
<td></td>
<td>Half transfer or transfer complete or transfer error on channel x</td>
<td>GIX</td>
<td>-</td>
</tr>
</tbody>
</table>

16.6  BDMA registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

The BDMA registers have to be accessed by words (32-bit).

16.6.1  BDMA interrupt status register (BDMA_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

Every status bit is cleared by hardware when the software sets the corresponding clear bit or the corresponding global clear bit CGIFx, in the BDMA_IFCR register.

<table>
<thead>
<tr>
<th>TEIF7</th>
<th>HTIF7</th>
<th>TCIF7</th>
<th>GIF7</th>
<th>TEIF6</th>
<th>HTIF6</th>
<th>TCIF6</th>
<th>GIF6</th>
<th>TEIF5</th>
<th>HTIF5</th>
<th>TCIF5</th>
<th>GIF5</th>
<th>TEIF4</th>
<th>HTIF4</th>
<th>TCIF4</th>
<th>GIF4</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TEIF3</td>
<td>HTIF3</td>
<td>TCIF3</td>
<td>GIF3</td>
<td>TEIF2</td>
<td>HTIF2</td>
<td>TCIF2</td>
<td>GIF2</td>
<td>TEIF1</td>
<td>HTIF1</td>
<td>TCIF1</td>
<td>GIF1</td>
<td>TEIF0</td>
<td>HTIF0</td>
<td>TCIF0</td>
<td>GIF0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  **TEIF7**: transfer error (TE) flag for channel 7
0: no TE event
1: a TE event occurred

Bit 30  **HTIF7**: half transfer (HT) flag for channel 7
0: no HT event
1: a HT event occurred

Bit 29  **TCIF7**: transfer complete (TC) flag for channel 7
0: no TC event
1: a TC event occurred

Bit 28  **GIF7**: global interrupt flag for channel 7
0: no TE, HT or TC event
1: a TE, HT or TC event occurred
Bit 27  **TEIF6**: transfer error (TE) flag for channel 6
0: no TE event
1: a TE event occurred

Bit 26  **HTIF6**: half transfer (HT) flag for channel 6
0: no HT event
1: a HT event occurred

Bit 25  **TCIF6**: transfer complete (TC) flag for channel 6
0: no TC event
1: a TC event occurred

Bit 24  **GIF6**: global interrupt flag for channel 6
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 23  **TEIF5**: transfer error (TE) flag for channel 5
0: no TE event
1: a TE event occurred

Bit 22  **HTIF5**: half transfer (HT) flag for channel 5
0: no HT event
1: a HT event occurred

Bit 21  **TCIF5**: transfer complete (TC) flag for channel 5
0: no TC event
1: a TC event occurred

Bit 20  **GIF5**: global interrupt flag for channel 5
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 19  **TEIF4**: transfer error (TE) flag for channel 4
0: no TE event
1: a TE event occurred

Bit 18  **HTIF4**: half transfer (HT) flag for channel 4
0: no HT event
1: a HT event occurred

Bit 17  **TCIF4**: transfer complete (TC) flag for channel 4
0: no TC event
1: a TC event occurred

Bit 16  **GIF4**: global interrupt flag for channel 4
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 15  **TEIF3**: transfer error (TE) flag for channel 3
0: no TE event
1: a TE event occurred

Bit 14  **HTIF3**: half transfer (HT) flag for channel 3
0: no HT event
1: a HT event occurred

Bit 13  **TCIF3**: transfer complete (TC) flag for channel 3
0: no TC event
1: a TC event occurred
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td><strong>GIF3</strong>: global interrupt flag for channel 3</td>
<td>no TE, HT or TC event</td>
<td>a TE, HT or TC event occurred</td>
</tr>
<tr>
<td>11</td>
<td><strong>TEIF2</strong>: transfer error (TE) flag for channel 2</td>
<td>no TE event</td>
<td>a TE event occurred</td>
</tr>
<tr>
<td>10</td>
<td><strong>HTIF2</strong>: half transfer (HT) flag for channel 2</td>
<td>no HT event</td>
<td>a HT event occurred</td>
</tr>
<tr>
<td>9</td>
<td><strong>TCIF2</strong>: transfer complete (TC) flag for channel 2</td>
<td>no TC event</td>
<td>a TC event occurred</td>
</tr>
<tr>
<td>8</td>
<td><strong>GIF2</strong>: global interrupt flag for channel 2</td>
<td>no TE, HT or TC event</td>
<td>a TE, HT or TC event occurred</td>
</tr>
<tr>
<td>7</td>
<td><strong>TEIF1</strong>: transfer error (TE) flag for channel 1</td>
<td>no TE event</td>
<td>a TE event occurred</td>
</tr>
<tr>
<td>6</td>
<td><strong>HTIF1</strong>: half transfer (HT) flag for channel 1</td>
<td>no HT event</td>
<td>a HT event occurred</td>
</tr>
<tr>
<td>5</td>
<td><strong>TCIF1</strong>: transfer complete (TC) flag for channel 1</td>
<td>no TC event</td>
<td>a TC event occurred</td>
</tr>
<tr>
<td>4</td>
<td><strong>GIF1</strong>: global interrupt flag for channel 1</td>
<td>no TE, HT or TC event</td>
<td>a TE, HT or TC event occurred</td>
</tr>
<tr>
<td>3</td>
<td><strong>TEIF0</strong>: transfer error (TE) flag for channel 0</td>
<td>no TE event</td>
<td>a TE event occurred</td>
</tr>
<tr>
<td>2</td>
<td><strong>HTIF0</strong>: half transfer (HT) flag for channel 0</td>
<td>no HT event</td>
<td>a HT event occurred</td>
</tr>
<tr>
<td>1</td>
<td><strong>TCIF0</strong>: transfer complete (TC) flag for channel 0</td>
<td>no TC event</td>
<td>a TC event occurred</td>
</tr>
<tr>
<td>0</td>
<td><strong>GIF0</strong>: global interrupt flag for channel 0</td>
<td>no TE, HT or TC event</td>
<td>a TE, HT or TC event occurred</td>
</tr>
</tbody>
</table>
### 16.6.2 BDMA interrupt flag clear register (BDMA_IFCR)

Address offset: 0x04
Reset value: 0x0000 0000

Setting the global clear bit CGFx of the channel x in this BDMA_IFCR register, causes the BDMA hardware to clear the corresponding GIFx bit and any individual flag among TEIFx, HTIFx, TCIFx, in the BDMA_ISR register.

Setting any individual clear bit among CTEIFx, CHTIFx, CTCIFx in this BDMA_IFCR register, causes the BDMA hardware to clear the corresponding individual flag and the global flag GIFx in the BDMA_ISR register, provided that none of the two other individual flags is set.

Writing 0 into any flag clear bit has no effect.

| Bit 31 | CTEIF7: transfer error flag clear for channel 7 |
| Bit 30 | CHTIF7: half transfer flag clear for channel 7 |
| Bit 29 | CTCIF7: transfer complete flag clear for channel 7 |
| Bit 28 | CGIF7: global interrupt flag clear for channel 7 |
| Bit 27 | CTEIF6: transfer error flag clear for channel 6 |
| Bit 26 | CHTIF6: half transfer flag clear for channel 6 |
| Bit 25 | CTCIF6: transfer complete flag clear for channel 6 |
| Bit 24 | CGIF6: global interrupt flag clear for channel 6 |
| Bit 23 | CTEIF5: transfer error flag clear for channel 5 |
| Bit 22 | CHTIF5: half transfer flag clear for channel 5 |
| Bit 21 | CTCIF5: transfer complete flag clear for channel 5 |
| Bit 20 | CGIF5: global interrupt flag clear for channel 5 |
| Bit 19 | CTEIF4: transfer error flag clear for channel 4 |
| Bit 18 | CHTIF4: half transfer flag clear for channel 4 |
| Bit 17 | CTCIF4: transfer complete flag clear for channel 4 |
| Bit 16 | CGIF4: global interrupt flag clear for channel 4 |
| Bit 15 | CTEIF3: transfer error flag clear for channel 3 |
| Bit 14 | CHTIF3: half transfer flag clear for channel 3 |
| Bit 13 | CTCIF3: transfer complete flag clear for channel 3 |
Bit 12 **CGIF3**: global interrupt flag clear for channel 3
Bit 11 **CTEIF2**: transfer error flag clear for channel 2
Bit 10 **CHTIF2**: half transfer flag clear for channel 2
Bit 9 **CTCIF2**: transfer complete flag clear for channel 2
Bit 8 **CGIF2**: global interrupt flag clear for channel 2
Bit 7 **CTEIF1**: transfer error flag clear for channel 1
Bit 6 **CHTIF1**: half transfer flag clear for channel 1
Bit 5 **CTCIF1**: transfer complete flag clear for channel 1
Bit 4 **CGIF1**: global interrupt flag clear for channel 0
Bit 3 **CTEIF0**: transfer error flag clear for channel 0
Bit 2 **CHTIF0**: half transfer flag clear for channel 0
Bit 1 **CTCIF0**: transfer complete flag clear for channel 0
Bit 0 **CGIF0**: global interrupt flag clear for channel 0

### 16.6.3 BDMA channel x configuration register (BDMA_CCRx)

Address offset: 0x08 + 0x14 * x, (x = 0 to 7)
Reset value: 0x0000 0000

The register fields/bits CT, DBM, MEM2MEM, PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, and DIR are read-only when EN = 1.

The states of MEM2MEM and CIRC bits must not be both high at the same time.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DBM</td>
<td>MEM2MEM</td>
<td>PL[1:0]</td>
<td>MSIZE[1:0]</td>
<td>PSIZE[1:0]</td>
<td>MINC</td>
<td>PINC</td>
<td>CIRC</td>
<td>DIR</td>
<td>TEIE</td>
<td>HTIE</td>
<td>TCIE</td>
<td>EN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:17 Reserved, must be kept at reset value.

Bit 16 CT: current target memory of DMA transfer in double-buffer mode
  This bit is toggled by hardware at the end of each channel transfer in double-buffer mode.
  0: memory 0 (addressed by the BDMA_CM0AR pointer)
  1: memory 1 (addressed by the BDMA_CM1AR pointer)
  Note: this bit is set and cleared by software.
  It must not be written when the channel is enabled (EN = 1).
  It is read-only when the channel is enabled (EN = 1).

Bit 15 DBM: double-buffer mode
  This bit must be set only in memory-to-peripheral and peripheral-to-memory transfers
  (MEM2MEM=0). The CIRC bit must also be set in double buffer mode.
  0: disabled (no memory address switch at the end of the BDMA transfer)
  1: enabled (memory address switched at the end of the BDMA transfer)
  Note: this bit is set and cleared by software.
  It must not be written when the channel is enabled (EN = 1).
  It is not read-only when the channel is enabled (EN = 1).

Bit 14 MEM2MEM: memory-to-memory mode
  0: disabled
  1: enabled
  Note: this bit is set and cleared by software.
  It must not be written when the channel is enabled (EN = 1).
  It is read-only when the channel is enabled (EN = 1).

Bits 13:12 PL[1:0]: priority level
  00: low
  01: medium
  10: high
  11: very high
  Note: this field is set and cleared by software.
  It must not be written when the channel is enabled (EN = 1).
  It is read-only when the channel is enabled (EN = 1).

Bits 11:10 MSIZE[1:0]: memory size
  Defines the data size of each DMA transfer to the identified memory.
  In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the
  memory destination if DIR = 0.
  In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the
  peripheral destination if DIR = 0.
  00: 8 bits
  01: 16 bits
  10: 32 bits
  11: reserved
  Note: this field is set and cleared by software.
  It must not be written when the channel is enabled (EN = 1).
  It is read-only when the channel is enabled (EN = 1).
Bits 9:8 **PSIZE[1:0]**: peripheral size

Defines the data size of each DMA transfer to the identified peripheral.
In memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.
In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.
00: 8 bits
01: 16 bits
10: 32 bits
11: reserved

Note: this field is set and cleared by software.

It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 7 **MINC**: memory increment mode

Defines the increment mode for each DMA transfer to the identified memory.
In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the memory destination if DIR = 0.
In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.
0: disabled
1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 6 **PINC**: peripheral increment mode

Defines the increment mode for each DMA transfer to the identified peripheral.
In memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.
In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.
0: disabled
1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 5 **CIRC**: circular mode

0: disabled
1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).
Bit 4  **DIR**: data transfer direction  
This bit must be set only in memory-to-peripheral and peripheral-to-memory modes.  
0: read from peripheral  
– Source attributes are defined by PSIZE and PINC, plus the BDMA_CPARx register.  
- This is still valid in a memory-to-memory mode.  
– Destination attributes are defined by MSIZE and MINC, plus the BDMA_CM0/1ARx  
register. This is still valid in a peripheral-to-peripheral mode.  
1: read from memory  
– Destination attributes are defined by PSIZE and PINC, plus the BDMA_CPARx  
register. This is still valid in a memory-to-memory mode.  
– Source attributes are defined by MSIZE and MINC, plus the BDMA_CM0/1ARx  
register. This is still valid in a peripheral-to-peripheral mode.  

*Note:* this bit is set and cleared by software.  
*It must not be written when the channel is enabled (EN = 1).*  
*It is read-only when the channel is enabled (EN = 1).*

Bit 3  **TEIE**: transfer error interrupt enable  
0: disabled  
1: enabled  

*Note:* this bit is set and cleared by software.  
*It must not be written when the channel is enabled (EN = 1).*  
*It is not read-only when the channel is enabled (EN = 1).*

Bit 2  **HTIE**: half transfer interrupt enable  
0: disabled  
1: enabled  

*Note:* this bit is set and cleared by software.  
*It must not be written when the channel is enabled (EN = 1).*  
*It is not read-only when the channel is enabled (EN = 1).*

Bit 1  **TCIE**: transfer complete interrupt enable  
0: disabled  
1: enabled  

*Note:* this bit is set and cleared by software.  
*It must not be written when the channel is enabled (EN = 1).*  
*It is not read-only when the channel is enabled (EN = 1).*

Bit 0  **EN**: channel enable  
When a channel transfer error occurs, this bit is cleared by hardware. It can not be set again  
by software (channel x re-activated) until the TEIFx bit of the BDMA_ISR register is cleared  
(by setting the CTEIFx bit of the BDMA_IFCR register).  
0: disabled  
1: enabled  

*Note:* this bit is set and cleared by software.
16.6.4 BDMA channel x number of data to transfer register (BDMA_CNDTRx)

Address offset: 0x0C + 0x14 * x, (x = 0 to 7)
Reset value: 0x0000 0000

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **NDT[15:0]**: number of data to transfer (0 to 2¹⁶ - 1)

This field is updated by hardware when the channel is enabled:

- It is decremented after each single BDMA 'read followed by write' transfer, indicating the remaining amount of data items to transfer.
- It is kept at zero when the programmed amount of data to transfer is reached, if the channel is not in circular mode (CIRC = 0 in the BDMA_CCRx register).
- It is reloaded automatically by the previously programmed value, when the transfer is complete, if the channel is in circular mode (CIRC = 1).

If this field is zero, no transfer can be served whatever the channel status (enabled or not).

**Note:** this field is set and cleared by software.

*It must not be written when the channel is enabled (EN = 1).*

*It is read-only when the channel is enabled (EN = 1).*

16.6.5 BDMA channel x peripheral address register (BDMA_CPARx)

Address offset: 0x10 + 0x14 * x, (x = 0 to 7)
Reset value: 0x0000 0000
### 16.6.6 BDMA channel x memory 0 address register (BDMA_CM0ARx)

Address offset: 0x14 + 0x14 * x, (x = 0 to 7)

Reset value: 0x0000 0000

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

Bits 31:0 **MA[31:0]**: peripheral address

- It contains the base address of the memory from/to which the data is read/written.
- When MSIZE[1:0] = 01 (16 bits), bit 0 of MA[31:0] is ignored. Access is automatically aligned to a half-word address.
- When MSIZE = 10 (32 bits), bits 1 and 0 of MA[31:0] are ignored. Access is automatically aligned to a word address.
- In memory-to-memory mode, this register identifies the memory source address if DIR = 1 and the memory destination address if DIR = 0.
- In peripheral-to-peripheral mode, this register identifies the peripheral source address if DIR = 1 and the peripheral destination address if DIR = 0.

**Note:** this register is set and cleared by software.

*It must not be written when the channel is enabled (EN = 1).*

*It is not read-only when the channel is enabled (EN = 1).*
16.6.7 BDMA channel x memory 1 address register (BDMA_CM1ARx)

Address offset: 0x18 + 0x14 * x, (x = 0 to 7)
Reset value: 0x0000 0000

Bits 31:0 MA[31:0]: peripheral address
It contains the base address of the memory from/to which the data is read/written.
When MSIZE[1:0] = 01 (16 bits), bit 0 of MA[31:0] is ignored. Access is automatically aligned to a half-word address.
When MSIZE = 10 (32 bits), bits 1 and 0 of MA[31:0] are ignored. Access is automatically aligned to a word address.
In memory-to-memory mode, this register identifies the memory source address if DIR = 1 and the memory destination address if DIR = 0.
In peripheral-to-peripheral mode, this register identifies the peripheral source address if DIR = 1 and the peripheral destination address if DIR = 0.
Note: this register is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

16.6.8 BDMA register map
The table below gives the BDMA register map and reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>MA[31:16]</th>
<th>MA[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>BDMA_ISR</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0x004</td>
<td>BDMA_IFCR</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0x008</td>
<td>BDMA_CCR0</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0x00C</td>
<td>BDMA_CNDTR0</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0x010</td>
<td>BDMA_CPAR0</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0x014</td>
<td>BDMA_CM0AR0</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0x018</td>
<td>BDMA_CM1AR0</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
## Table 99. BDMA register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Reset value</th>
<th>CT</th>
<th>DBM</th>
<th>MEM2MEM</th>
<th>PL[1:0]</th>
<th>MSIZE[1:0]</th>
<th>PSIZE[1:0]</th>
<th>MINC</th>
<th>INC</th>
<th>CIRC</th>
<th>DIR</th>
<th>TEIE</th>
<th>HTIE</th>
<th>TCIE</th>
<th>TCE</th>
<th>EN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01C</td>
<td>BDMA_CCR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x020</td>
<td>BDMA_CNDTR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x024</td>
<td>BDMA_CPAR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x028</td>
<td>BDMA_CM0AR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x02C</td>
<td>BDMA_CM1AR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x030</td>
<td>BDMA_CCR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x034</td>
<td>BDMA_CNDTR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x038</td>
<td>BDMA_CPAR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x03C</td>
<td>BDMA_CM0AR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x040</td>
<td>BDMA_CM1AR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x044</td>
<td>BDMA_CCR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x048</td>
<td>BDMA_CNDTR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x04C</td>
<td>BDMA_CPAR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x050</td>
<td>BDMA_CM0AR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x054</td>
<td>BDMA_CM1AR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x058</td>
<td>BDMA_CCR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x05C</td>
<td>BDMA_CNDTR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x060</td>
<td>BDMA_CPAR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x064</td>
<td>BDMA_CM0AR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x068</td>
<td>BDMA_CM1AR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Refer to Section 2.3 on page 128 for the register boundary addresses.
17 DMA request multiplexer (DMAMUX)

17.1 Introduction

A peripheral indicates a request for DMA transfer by setting its DMA request signal. The DMA request is pending until it is served by the DMA controller that generates a DMA acknowledge signal, and the corresponding DMA request signal is deasserted.

In this document, the set of control signals required for the DMA request/acknowledge protocol is not explicitly shown or described, and it is referred to as DMA request line.

The DMAMUX request multiplexer enables routing a DMA request line between the peripherals and the DMA controller of the product. The routing function is ensured by a programmable multi-channel DMA request line multiplexer. Each channel selects a unique DMA request line, unconditionally or synchronously with events from its DMAMUX synchronization inputs. The DMAMUX may also be used as a DMA request generator from programmable events on its input trigger signals.

The number of DMAMUX instances and their main characteristics are specified in Section 17.3.1.

The assignment of DMAMUX request multiplexer inputs to the DMA request lines from peripherals and to the DMAMUX request generator outputs, the assignment of DMAMUX request multiplexer outputs to DMA controller channels, and the assignment of DMAMUX synchronizations and trigger inputs to internal and external signals depend on the product implementation, and are detailed in Section 17.3.2.
17.2 **DMAMUX main features**

- Up to 16-channel programmable DMA request line multiplexer output
- Up to 8-channel DMA request generator
- Up to 32 trigger inputs to DMA request generator
- Up to 16 synchronization inputs
- Per DMA request generator channel:
  - DMA request trigger input selector
  - DMA request counter
  - Event overrun flag for selected DMA request trigger input
- Per DMA request line multiplexer channel output:
  - Up to 107 input DMA request lines from peripherals
  - One DMA request line output
  - Synchronization input selector
  - DMA request counter
  - Event overrun flag for selected synchronization input
  - One event output, for DMA request chaining

17.3 **DMAMUX implementation**

17.3.1 **DMAMUX1 and DMAMUX2 instantiation**

The product integrates two instances of DMA request multiplexer:

- DMAMUX1 for DMA1 and DMA2 (CD domain)
- DMAMUX2 for BDMA (SRD domain)

DMAMUX1 and DMAMUX2 are instantiated with the hardware configuration parameters listed in the following table.

<table>
<thead>
<tr>
<th>Feature</th>
<th>DMAMUX1</th>
<th>DMAMUX2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of DMAMUX output request channels</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Number of DMAMUX request generator channels</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Number of DMAMUX request trigger inputs</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td>Number of DMAMUX synchronization inputs</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>Number of DMAMUX peripheral request inputs</td>
<td>101</td>
<td>8</td>
</tr>
</tbody>
</table>

17.3.2 **DMAMUX1 mapping**

The mapping of resources to DMAMUX is hardwired.

DMAMUX1 is used with the DMA1 and DMA2 in CD domain:

- DMAMUX1 channels 0 to 7 are connected to DMA1 channels 0 to 7
- DMAMUX1 channels 8 to 15 are connected to DMA2 channels 0 to 7
### Table 101. DMAMUX1: assignment of multiplexer inputs to resources

<table>
<thead>
<tr>
<th>DMA request MUX input</th>
<th>Resource</th>
<th>DMA request MUX input</th>
<th>Resource</th>
<th>DMA request MUX input</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dmamux1_req_gen0</td>
<td>43</td>
<td>USART2_RX</td>
<td>85</td>
<td>SPI5_RX</td>
</tr>
<tr>
<td>2</td>
<td>dmamux1_req_gen1</td>
<td>44</td>
<td>USART2_TX</td>
<td>86</td>
<td>SPI5_TX</td>
</tr>
<tr>
<td>3</td>
<td>dmamux1_req_gen2</td>
<td>45</td>
<td>USART3_RX</td>
<td>87</td>
<td>SAI1_A</td>
</tr>
<tr>
<td>4</td>
<td>dmamux1_req_gen3</td>
<td>46</td>
<td>USART3_TX</td>
<td>88</td>
<td>SAI1_B</td>
</tr>
<tr>
<td>5</td>
<td>dmamux1_req_gen4</td>
<td>47</td>
<td>TIM8_CH1</td>
<td>89</td>
<td>SAI2_A</td>
</tr>
<tr>
<td>6</td>
<td>dmamux1_req_gen5</td>
<td>48</td>
<td>TIM8_CH2</td>
<td>90</td>
<td>SAI2_B</td>
</tr>
<tr>
<td>7</td>
<td>dmamux1_req_gen6</td>
<td>49</td>
<td>TIM8_CH3</td>
<td>91</td>
<td>SWPMI_RX</td>
</tr>
<tr>
<td>8</td>
<td>dmamux1_req_gen7</td>
<td>50</td>
<td>TIM8_CH4</td>
<td>92</td>
<td>SWPMI_TX</td>
</tr>
<tr>
<td>9</td>
<td>ADC1</td>
<td>51</td>
<td>TIM8_UP</td>
<td>93</td>
<td>SPDIFRX_DT</td>
</tr>
<tr>
<td>10</td>
<td>ADC2</td>
<td>52</td>
<td>TIM8_TRIG</td>
<td>94</td>
<td>SPDIFRX_CS</td>
</tr>
<tr>
<td>11</td>
<td>TIM1_CH1</td>
<td>53</td>
<td>TIM8_COM</td>
<td>95</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>TIM1_CH2</td>
<td>54</td>
<td>Reserved</td>
<td>96</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>TIM1_CH3</td>
<td>55</td>
<td>TIM5_CH1</td>
<td>97</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>TIM1_CH4</td>
<td>56</td>
<td>TIM5_CH2</td>
<td>98</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>TIM1_UP</td>
<td>57</td>
<td>TIM5_CH3</td>
<td>99</td>
<td>Reserved</td>
</tr>
<tr>
<td>16</td>
<td>TIM1_TRIG</td>
<td>58</td>
<td>TIM5_CH4</td>
<td>100</td>
<td>Reserved</td>
</tr>
<tr>
<td>17</td>
<td>TIM1_COM</td>
<td>59</td>
<td>TIM5_UP</td>
<td>101</td>
<td>DFSDM1_dma0</td>
</tr>
<tr>
<td>18</td>
<td>TIM2_CH1</td>
<td>60</td>
<td>TIM5_TRIG</td>
<td>102</td>
<td>DFSDM1_dma1</td>
</tr>
<tr>
<td>19</td>
<td>TIM2_CH2</td>
<td>61</td>
<td>SPI3_RX</td>
<td>103</td>
<td>DFSDM1_dma2</td>
</tr>
<tr>
<td>20</td>
<td>TIM2_CH3</td>
<td>62</td>
<td>SPI3_TX</td>
<td>104</td>
<td>DFSDM1_dma3</td>
</tr>
<tr>
<td>21</td>
<td>TIM2_CH4</td>
<td>63</td>
<td>UART4_RX</td>
<td>105</td>
<td>TIM15_CH1</td>
</tr>
<tr>
<td>22</td>
<td>TIM2_UP</td>
<td>64</td>
<td>UART4_TX</td>
<td>106</td>
<td>TIM15_UP</td>
</tr>
<tr>
<td>23</td>
<td>TIM3_CH1</td>
<td>65</td>
<td>UART5_RX</td>
<td>107</td>
<td>TIM15_TRIG</td>
</tr>
<tr>
<td>24</td>
<td>TIM3_CH2</td>
<td>66</td>
<td>UART5_TX</td>
<td>108</td>
<td>TIM15_COM</td>
</tr>
<tr>
<td>25</td>
<td>TIM3_CH3</td>
<td>67</td>
<td>DAC1_out1</td>
<td>109</td>
<td>TIM16_CH1</td>
</tr>
<tr>
<td>26</td>
<td>TIM3_CH4</td>
<td>68</td>
<td>DAC1_out2</td>
<td>110</td>
<td>TIM16_UP</td>
</tr>
<tr>
<td>27</td>
<td>TIM3_UP</td>
<td>69</td>
<td>TIM6_UP</td>
<td>111</td>
<td>TIM17_CH1</td>
</tr>
<tr>
<td>28</td>
<td>TIM3_TRIG</td>
<td>70</td>
<td>TIM7_UP</td>
<td>112</td>
<td>TIM17_UP</td>
</tr>
<tr>
<td>29</td>
<td>TIM4_CH1</td>
<td>71</td>
<td>USART6_RX</td>
<td>113</td>
<td>Reserved</td>
</tr>
<tr>
<td>30</td>
<td>TIM4_CH2</td>
<td>72</td>
<td>USART6_TX</td>
<td>114</td>
<td>Reserved</td>
</tr>
<tr>
<td>31</td>
<td>TIM4_CH3</td>
<td>73</td>
<td>I2C3_RX</td>
<td>115</td>
<td>Reserved</td>
</tr>
<tr>
<td>32</td>
<td>TIM4_UP</td>
<td>74</td>
<td>I2C3_TX</td>
<td>116</td>
<td>UART9_RX</td>
</tr>
<tr>
<td>33</td>
<td>I2C1_RX</td>
<td>75</td>
<td>DCMI_PSSI</td>
<td>117</td>
<td>UART9_TX</td>
</tr>
<tr>
<td>34</td>
<td>I2C1_TX</td>
<td>76</td>
<td>CRYP_IN</td>
<td>118</td>
<td>USART10_RX</td>
</tr>
<tr>
<td>35</td>
<td>I2C2_RX</td>
<td>77</td>
<td>CRYP_OUT</td>
<td>119</td>
<td>USART10_TX</td>
</tr>
<tr>
<td>36</td>
<td>I2C2_TX</td>
<td>78</td>
<td>HASH_IN</td>
<td>120</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
Table 101. DMAMUX1: assignment of multiplexer inputs to resources (continued)

<table>
<thead>
<tr>
<th>DMA request MUX input</th>
<th>Resource</th>
<th>DMA request MUX input</th>
<th>Resource</th>
<th>DMA request MUX input</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>SPI1_RX</td>
<td>79</td>
<td>UART7_RX</td>
<td>121</td>
<td>Reserved</td>
</tr>
<tr>
<td>38</td>
<td>SPI1_TX</td>
<td>80</td>
<td>UART7_TX</td>
<td>122</td>
<td>Reserved</td>
</tr>
<tr>
<td>39</td>
<td>SPI2_RX</td>
<td>81</td>
<td>UART8_RX</td>
<td>123</td>
<td>Reserved</td>
</tr>
<tr>
<td>40</td>
<td>SPI2_TX</td>
<td>82</td>
<td>UART8_TX</td>
<td>124</td>
<td>Reserved</td>
</tr>
<tr>
<td>41</td>
<td>USART1_RX</td>
<td>83</td>
<td>SPI4_RX</td>
<td>125</td>
<td>Reserved</td>
</tr>
<tr>
<td>42</td>
<td>USART1_TX</td>
<td>84</td>
<td>SPI4_TX</td>
<td>127</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 102. DMAMUX1: assignment of trigger inputs to resources

<table>
<thead>
<tr>
<th>Trigger input</th>
<th>Resource</th>
<th>Trigger input</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DMAMUX1_evt0</td>
<td>4</td>
<td>LPTIMER2_out</td>
</tr>
<tr>
<td>1</td>
<td>DMAMUX1_evt1</td>
<td>5</td>
<td>LPTIMER3_out</td>
</tr>
<tr>
<td>2</td>
<td>DMAMUX1_evt2</td>
<td>6</td>
<td>extit0</td>
</tr>
<tr>
<td>3</td>
<td>LPTIMER1_out</td>
<td>7</td>
<td>TIM12_TRGO</td>
</tr>
</tbody>
</table>

Table 103. DMAMUX1: assignment of synchronization inputs to resources

<table>
<thead>
<tr>
<th>Sync. input</th>
<th>Resource</th>
<th>Sync. input</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DMAMUX1_evt0</td>
<td>4</td>
<td>LPTIMER2_out</td>
</tr>
<tr>
<td>1</td>
<td>DMAMUX1_evt1</td>
<td>5</td>
<td>LPTIMER3_out</td>
</tr>
<tr>
<td>2</td>
<td>DMAMUX1_evt2</td>
<td>6</td>
<td>extit0</td>
</tr>
<tr>
<td>3</td>
<td>LPTIMER1_out</td>
<td>7</td>
<td>TIM12_TRGO</td>
</tr>
</tbody>
</table>

17.3.3 DMAMUX2 mapping

DMAMUX2 channels 0 to 7 are connected to BDMA channels 0 to 7.

Table 104. DMAMUX2: assignment of multiplexer inputs to resources

<table>
<thead>
<tr>
<th>DMA request MUX input</th>
<th>Resource</th>
<th>DMA request MUX input</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dmasux2_req_gen0</td>
<td>17</td>
<td>DAC2</td>
</tr>
<tr>
<td>2</td>
<td>dmasux2_req_gen1</td>
<td>18</td>
<td>DFSDM2_dma0</td>
</tr>
<tr>
<td>3</td>
<td>dmasux2_req_gen2</td>
<td>19</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>dmasux2_req_gen3</td>
<td>20</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>dmasux2_req_gen4</td>
<td>21</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>dmasux2_req_gen5</td>
<td>22</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>dmasux2_req_gen6</td>
<td>23</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
### Table 104. DMAMUX2: assignment of multiplexer inputs to resources (continued)

<table>
<thead>
<tr>
<th>DMA request MUX input</th>
<th>Resource</th>
<th>DMA request MUX input</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>dmamux2_req_gen7</td>
<td>24</td>
<td>Reserved</td>
</tr>
<tr>
<td>9</td>
<td>LP UART1_RX</td>
<td>25</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>LP UART1_TX</td>
<td>26</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>SPI6_RX</td>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>SPI6_TX</td>
<td>28</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>I2C4_RX</td>
<td>29</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>I2C4_TX</td>
<td>30</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td>31</td>
<td>Reserved</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Trigger input</td>
<td>Resource</td>
<td>Trigger input</td>
<td>Resource</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------</td>
<td>---------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>0</td>
<td>DMAMUX2_evt0</td>
<td>16</td>
<td>Spi6_it_async</td>
</tr>
<tr>
<td>1</td>
<td>DMAMUX2_evt1</td>
<td>17</td>
<td>Comp1_out</td>
</tr>
<tr>
<td>2</td>
<td>DMAMUX2_evt2</td>
<td>18</td>
<td>Comp2_out</td>
</tr>
<tr>
<td>3</td>
<td>DMAMUX2_evt3</td>
<td>19</td>
<td>RTC_wkup</td>
</tr>
<tr>
<td>4</td>
<td>DMAMUX2_evt4</td>
<td>20</td>
<td>Syscfg_exti0_mux</td>
</tr>
<tr>
<td>5</td>
<td>DMAMUX2_evt5</td>
<td>21</td>
<td>Syscfg_exti2_mux</td>
</tr>
<tr>
<td>6</td>
<td>DMAMUX2_evt6</td>
<td>22</td>
<td>I2c4_it_event</td>
</tr>
<tr>
<td>7</td>
<td>Lpuart1_it_R_WUP_ASYNC</td>
<td>23</td>
<td>Spi6_it</td>
</tr>
<tr>
<td>8</td>
<td>Lpuart1_it_T_WUP_ASYNC</td>
<td>24</td>
<td>Lpuart1_it_T</td>
</tr>
<tr>
<td>9</td>
<td>Lptim2_ait</td>
<td>25</td>
<td>Lpuart1_it_R</td>
</tr>
<tr>
<td>10</td>
<td>Lptim2_out</td>
<td>26</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>Lptim3_ait</td>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>Lptim3_out</td>
<td>28</td>
<td>BDMA2_ch0_it</td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
<td>29</td>
<td>BDMA2_ch1_it</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td>30</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>I2c4_it_async</td>
<td>31</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sync input</th>
<th>Resource</th>
<th>Sync input</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>dmamux2_ch0_evt</td>
<td>8</td>
<td>Lptim2_out</td>
</tr>
<tr>
<td>1</td>
<td>dmamux2_ch1_evt</td>
<td>9</td>
<td>Lptim3_out</td>
</tr>
<tr>
<td>2</td>
<td>dmamux2_ch2_evt</td>
<td>10</td>
<td>I2c4_it_async</td>
</tr>
<tr>
<td>3</td>
<td>dmamux2_ch3_evt</td>
<td>11</td>
<td>Spi6_it_async</td>
</tr>
<tr>
<td>4</td>
<td>dmamux2_ch4_evt</td>
<td>12</td>
<td>Comp1_out</td>
</tr>
<tr>
<td>5</td>
<td>dmamux2_ch5_evt</td>
<td>13</td>
<td>RTC_wkup</td>
</tr>
<tr>
<td>6</td>
<td>Lpuart1_it_R_WUP_ASYNC</td>
<td>14</td>
<td>Syscfg_exti0_mux</td>
</tr>
<tr>
<td>7</td>
<td>Lpuart1_it_T_WUP_ASYNC</td>
<td>15</td>
<td>Syscfg_exti2_mux</td>
</tr>
</tbody>
</table>
17.4 DMAMUX functional description

17.4.1 DMAMUX block diagram

*Figure 87* shows the DMAMUX block diagram.

**Figure 87. DMAMUX block diagram**

DMAMUX features two main sub-blocks: the request line multiplexer and the request line generator.

The implementation assigns:
- DMAMUX request multiplexer sub-block inputs (dmamux_reqx) from peripherals (dmamux_req_inx) and from channels of the DMAMUX request generator sub-block (dmamux_req_genx)
- DMAMUX request outputs to channels of DMA controller (dmamux_req_outx)
- Internal or external signals to DMA request trigger inputs (dmamux_trgx)
- Internal or external signals to synchronization inputs (dmamux_syncx)
17.4.2 DMAMUX signals

*Table 107 lists the DMAMUX signals.*

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dmamux_hclk</td>
<td>DMAMUX AHB clock</td>
</tr>
<tr>
<td>dmamux_req_inx</td>
<td>DMAMUX DMA request line inputs from peripherals</td>
</tr>
<tr>
<td>dmamux_trgx</td>
<td>DMAMUX DMA request triggers inputs (to request generator sub-block)</td>
</tr>
<tr>
<td>dmamux_req_genx</td>
<td>DMAMUX request generator sub-block channels outputs</td>
</tr>
<tr>
<td>dmamux_reqx</td>
<td>DMAMUX request multiplexer sub-block inputs (from peripheral requests and request generator channels)</td>
</tr>
<tr>
<td>dmamux_syncx</td>
<td>DMAMUX synchronization inputs (to request multiplexer sub-block)</td>
</tr>
<tr>
<td>dmamux_req_outx</td>
<td>DMAMUX requests outputs (to DMA controller)</td>
</tr>
<tr>
<td>dmamux_evtx</td>
<td>DMAMUX events outputs</td>
</tr>
<tr>
<td>dmamux_ovr_it</td>
<td>DMAMUX overrun interrupts</td>
</tr>
</tbody>
</table>

17.4.3 DMAMUX channels

A DMAMUX channel is a DMAMUX request multiplexer channel that may include, depending on the selected input of the request multiplexer, an additional DMAMUX request generator channel.

A DMAMUX request multiplexer channel is connected and dedicated to one single channel of DMA controller.

*Channel configuration procedure*

Follow the sequence below to configure both a DMAMUX x channel and the related DMA channel y:

1. Set and configure completely the DMA channel y, except enabling the channel y.
2. Set and configure completely the related DMAMUX y channel.
3. Last, activate the DMA channel y by setting the EN bit in the DMA y channel register.

17.4.4 DMAMUX request line multiplexer

The DMAMUX request multiplexer with its multiple channels ensures the actual routing of DMA request/acknowledge control signals, named DMA request lines.

Each DMA request line is connected in parallel to all the channels of the DMAMUX request line multiplexer.

A DMA request is sourced either from the peripherals or from the DMAMUX request generator.

The DMAMUX request line multiplexer channel x selects the DMA request line number as configured by the DMA_REQ_ID field in the DMAMUX_CxCR register.
Note: The null value in the field DMAREQ_ID corresponds to no DMA request line selected. It is not allowed to configure a same non-null DMAREQ_ID to two different channels of the DMAMUX request line multiplexer.

On top of the DMA request selection, the synchronization mode and/or the event generation may be configured and enabled, if required.

Synchronization mode and channel event generation

Each DMAMUX request line multiplexer channel x can be individually synchronized by setting the synchronization enable (SE) bit in the DMAMUX_CxCR register.

DMAMUX has multiple synchronization inputs. The synchronization inputs are connected in parallel to all the channels of the request multiplexer.

The synchronization input is selected via the SYNC_ID field in the DMAMUX_CxCR register of a given channel x.

When a channel is in this synchronization mode, the selected input DMA request line is propagated to the multiplexer channel output, once is detected a programmable rising/falling edge on the selected input synchronization signal, via the SPOL[1:0] field of the DMAMUX_CxCR register.

Additionally, there is a programmable DMA request counter, internally to the DMAMUX request multiplexer, which may be used for the channel request output generation and also possibly for an event generation. An event generation on the channel x output is enabled through the EGE bit (event generation enable) of the DMAMUX_CxCR register.

As shown in Figure 89, upon the detected edge of the synchronization input, the pending selected input DMA request line is connected to the DMAMUX multiplexer channel x output.

Note: If a synchronization event occurs while there is no pending selected input DMA request line, it is discarded. The following asserted input request lines is not connected to the DMAMUX multiplexer channel output until a synchronization event occurs again.

From this point on, each time the connected DMAMUX request is served by the DMA controller (a served request is deasserted), the DMAMUX request counter is decremented. At its underrun, the DMA request counter is automatically loaded with the value in NBREQ field of the DMAMUX_CxCR register and the input DMA request line is disconnected from the multiplexer channel x output.

Thus, the number of DMA requests transferred to the multiplexer channel x output following a detected synchronization event, is equal to the value in NBREQ field, plus one.

Note: The NBREQ field value shall only be written by software when both synchronization enable bit SE and event generation enable EGE bit of the corresponding multiplexer channel x are disabled.
If EGE is enabled, the multiplexer channel generates a channel event, as a pulse of one AHB clock cycle, when its DMA request counter is automatically reloaded with the value of the programmed NBREQ field, as shown in Figure 88 and Figure 89.
Note: If EGE is enabled and NBREQ = 0, an event is generated after each served DMA request.

Note: A synchronization event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.

Upon writing into DMAMUX_CxCR register, the synchronization events are masked during three AHB clock cycles.

Synchronization overrun and interrupt

If a new synchronization event occurs before the request counter underrun (the internal request counter programmed via the NBREQ field of the DMAMUX_CxCR register), the synchronization overrun flag bit SOFx is set in the DMAMUX_CSR status register.

Note: The request multiplexer channel x synchronization must be disabled (DMAMUX_CxCR.SE = 0) at the completion of the use of the related channel of the DMA controller. Else, upon a new detected synchronization event, there is a synchronization overrun due to the absence of a DMA acknowledge (that is, no served request) received from the DMA controller.

The overrun flag SOFx is reset by setting the associated clear synchronization overrun flag bit CSOFx in the DMAMUX_CFR register.

Setting the synchronization overrun flag generates an interrupt if the synchronization overrun interrupt enable bit SOIE is set in the DMAMUX_CxCR register.

17.4.5 DMAMUX request generator

The DMAMUX request generator produces DMA requests following trigger events on its DMA request trigger inputs.

The DMAMUX request generator has multiple channels. DMA request trigger inputs are connected in parallel to all channels.

The outputs of DMAMUX request generator channels are inputs to the DMAMUX request line multiplexer.

Each DMAMUX request generator channel x has an enable bit GE (generator enable) in the corresponding DMAMUX_RGxCR register.

The DMA request trigger input for the DMAMUX request generator channel x is selected through the SIG_ID (trigger signal ID) field in the corresponding DMAMUX_RGxCR register.

Trigger events on a DMA request trigger input can be rising edge, falling edge or either edge. The active edge is selected through the GPOL (generator polarity) field in the corresponding DMAMUX_RGxCR register.

Upon the trigger event, the corresponding generator channel starts generating DMA requests on its output. Each time the DMAMUX generated request is served by the connected DMA controller (a served request is deasserted), a built-in (inside the DMAMUX request generator) DMA request counter is decremented. At its underrun, the request generator channel stops generating DMA requests and the DMA request counter is automatically reloaded to its programmed value upon the next trigger event.

Thus, the number of DMA requests generated after the trigger event is GNBREQ + 1.
Note: The GNBREQ field value must be written by software only when the enable GE bit of the corresponding generator channel x is disabled.

A trigger event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.

Upon writing into DMAMUX_RGxCR register, the trigger events are masked during three AHB clock cycles.

Trigger overrun and interrupt

If a new DMA request trigger event occurs before the DMAMUX request generator counter underrun (the internal counter programmed via the GNBREQ field of the DMAMUX_RGxCR register), and if the request generator channel x was enabled via GE, then the request trigger event overrun flag bit OFx is asserted by the hardware in the status DMAMUX_RGSR register.

Note: The request generator channel x must be disabled (DMAMUX_RGxCR.GE = 0) at the completion of the usage of the related channel of the DMA controller. Else, upon a new detected trigger event, there is a trigger overrun due to the absence of an acknowledge (that is, no served request) received from the DMA.

The overrun flag OFx is reset by setting the associated clear overrun flag bit COFx in the DMAMUX_RGCFR register.

Setting the DMAMUX request trigger overrun flag generates an interrupt if the DMA request trigger event overrun interrupt enable bit OIE is set in the DMAMUX_RGxCR register.

17.5 DMAMUX interrupts

An interrupt can be generated upon:
- a synchronization event overrun in each DMA request line multiplexer channel
- a trigger event overrun in each DMA request generator channel

For each case, per-channel individual interrupt enable, status and clear flag register bits are available.

<table>
<thead>
<tr>
<th>Interrupt signal</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Clear bit</th>
<th>Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>dmamuxovr_it</td>
<td>Synchronization event overrun on channel x of the DMAMUX request line multiplexer</td>
<td>SOFx</td>
<td>CSOFx</td>
<td>SOIE</td>
</tr>
<tr>
<td></td>
<td>Trigger event overrun on channel x of the DMAMUX request generator</td>
<td>OFx</td>
<td>COFx</td>
<td>OIE</td>
</tr>
</tbody>
</table>

Table 108. DMAMUX interrupts
17.6 **DMAMUX registers**

Refer to the table containing register boundary addresses for the DMAMUX1 and DMAMUX2 base address.

DMAMUX registers may be accessed per (8-bit) byte, (16-bit) half-word, or (32-bit) word. The address must be aligned with the data size.

17.6.1 **DMAMUX1 request line multiplexer channel x configuration register (DMAMUX1_CxCR)**

Address offset: 0x000 + 0x04 * x (x = 0 to 15)

Reset value: 0x0000 0000

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

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

Bits 26:24 **SYNC_ID[2:0]**: Synchronization identification

Selects the synchronization input (see Table 101: DMAMUX1: assignment of multiplexer inputs to resources and Table 106: DMAMUX2: assignment of synchronization inputs to resources).

Bits 23:19 **NBREQ[4:0]**: Number of DMA requests minus 1 to forward

Defines the number of DMA requests to forward to the DMA controller after a synchronization event, and/or the number of DMA requests before an output event is generated.

This field shall only be written when both SE and EGE bits are low.

Bits 18:17 **SPOL[1:0]**: Synchronization polarity

Defines the edge polarity of the selected synchronization input:

00: no event, i.e. no synchronization nor detection.
01: rising edge
10: falling edge
11: rising and falling edge

Bit 16 **SE**: Synchronization enable

0: synchronization disabled
1: synchronization enabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **EGE**: Event generation enable

0: event generation disabled
1: event generation enabled

Bit 8 **SOIE**: Synchronization overrun interrupt enable

0: interrupt disabled
1: interrupt enabled
17.6.2 **DMAMUX2 request line multiplexer channel x configuration register (DMAMUX2_CxCR)**

Address offset: 0x000 + 0x04 * x, where x = 0 to 7

Reset value: 0x0000 0000

| Bit 31:28 | Reserved, must be kept at reset value. |
| Bit 27:24 | **SYNC_ID[3:0]**: Synchronization identification |
| Selects the synchronization input (see ) |

| Bit 23:19 | **NBREQ[4:0]**: Number of DMA requests minus 1 to forward |
| Defines the number of DMA requests to forward to the DMA controller after a synchronization event, and/or the number of DMA requests before an output event is generated. |
| This field shall only be written when both SE and EGE bits are low. |

| Bit 18:17 | **SPOL[1:0]**: Synchronization polarity |
| Defines the edge polarity of the selected synchronization input: |
| 00: no event, i.e. no synchronization nor detection. |
| 01: rising edge |
| 10: falling edge |
| 11: rising and falling edges |

| Bit 16 | **SE**: Synchronization enable |
| 0: synchronization disabled |
| 1: synchronization enabled |

| Bit 15:10 | Reserved, must be kept at reset value. |

| Bit 9 | **EGE**: Event generation enable |
| 0: event generation disabled |
| 1: event generation enabled |

| Bit 8 | **SOIE**: Synchronization overrun interrupt enable |
| 0: interrupt disabled |
| 1: interrupt enabled |

| Bit 7 | Reserved, must be kept at reset value. |

| Bit 6:0 | **DMAREQ_ID[6:0]**: DMA request identification |
| Selects the input DMA request. (see the DMAMUX table about assignments of multiplexer inputs to resources). |
### 17.6.3 DMAMUX1 request line multiplexer interrupt channel status register (DMAMUX1_CSR)

Address offset: 0x080

Reset value: 0x0000 0000

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

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

<table>
<thead>
<tr>
<th>SOF15</th>
<th>SOF14</th>
<th>SOF13</th>
<th>SOF12</th>
<th>SOF11</th>
<th>SOF10</th>
<th>SOF9</th>
<th>SOF8</th>
<th>SOF7</th>
<th>SOF6</th>
<th>SOF5</th>
<th>SOF4</th>
<th>SOF3</th>
<th>SOF2</th>
<th>SOF1</th>
<th>SOF0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SOF[15:0]**: Synchronization overrun event flag

- The flag is set when a synchronization event occurs on a DMA request line multiplexer channel x, while the DMA request counter value is lower than NBREQ.
- The flag is cleared by writing 1 to the corresponding CSOFx bit in DMAMUX_CFR register.
- For DMAMUX2_CFR bits 15:8 are reserved.

### 17.6.4 DMAMUX2 request line multiplexer interrupt channel status register (DMAMUX2_CSR)

Address offset: 0x080

Reset value: 0x0000 0000

This register shall be accessed at bit level by a non-secure or secure read, according to the secure mode of the considered DMAMUX2 request line multiplexer channel x, depending on the secure control bit of the connected DMA controller channel y, and considering that the DMAMUX2 x channel output is connected to the y channel of the DMA (refer to the DMAMUX2 mapping implementation section).

This register shall be accessed at bit level by an unprivileged or privileged read, according to the privileged mode of the considered DMAMUX2 request line multiplexer channel x, depending on the privileged control bit of the connected DMA controller channel y, and considering that the DMAMUX2 x channel output is connected to the y channel of the DMA (refer to the DMAMUX2 mapping implementation section).
17.6.5 DMAMUX1 request line multiplexer interrupt clear flag register (DMAMUX1_CFR)

Address offset: 0x084
Reset value: 0x0000 0000

| Bits 31:16 Reserved, must be kept at reset value. |

| Bits 7:0 SOF[7:0]: Synchronization overrun event flag |

The flag is set when a new synchronization event occurs on a DMA request line multiplexer channel x before the request counter underrun (the internal request counter programmed via the NBREQ field of the DMAMUX_CxCR register).

The flag is cleared by writing 1 to the corresponding CSOFx bit in DMAMUX2_CFR register.

17.6.6 DMAMUX2 request line multiplexer interrupt clear flag register (DMAMUX2_CFR)

Address offset: 0x084
Reset value: 0x0000 0000

This register shall be written at bit level by a non-secure or secure write, according to the secure mode of the considered DMAMUX request line multiplexer channel x, depending on the secure control bit of the connected DMA controller channel y, and considering that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMUX mapping implementation section).

This register shall be written at bit level by an unprivileged or privileged write, according to the privileged mode of the considered DMAMUX request line multiplexer channel x, depending on the privileged control bit of the connected DMA controller channel y, and considering that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMUX mapping implementation section).
17.6.7 DMAMUX1 request generator channel x configuration register (DMAMUX1_RGxCR)

Address offset: 0x100 + 0x04 * x (x = 0 to 7)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bits 7:0 | CSOF[7:0]: Clear synchronization overrun event flag  
Writing 1 in each bit clears the corresponding overrun flag SOFx in the DMAMUX2_CSR register. |

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

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:19 GNBREQ[4:0]: Number of DMA requests to be generated (minus 1)
Defines the number of DMA requests to be generated after a trigger event. The actual number of generated DMA requests is GNBREQ +1.

Note: This field shall only be written when GE bit is disabled.

Bits 18:17 GPOL[1:0]: DMA request generator trigger polarity
Defines the edge polarity of the selected trigger input
00: no event. I.e. none trigger detection nor generation.
01: rising edge
10: falling edge
11: rising and falling edge

Bit 16 GE: DMA request generator channel x enable
0: DMA request generator channel x disabled
1: DMA request generator channel x enabled

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

Bit 8 OIE: Trigger overrun interrupt enable
0: interrupt on a trigger overrun event occurrence is disabled
1: interrupt on a trigger overrun event occurrence is enabled

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

Bits 2:0 SIG_ID[2:0]: Signal identification
Selects the DMA request trigger input used for the channel x of the DMA request generator

17.6.8 DMAMUX2 request generator channel x configuration register (DMAMUX2_RGxCR)

Address offset: 0x100 + 0x04 * x (x = 0 to 7)
Reset value: 0x0000 0000
This register shall be written by a non-secure or secure write, according to the secure mode of the considered DMAMUX request line multiplexer channel y it is assigned to, and considering that the DMAMUX request generator x channel output is selected by the y channel of the DMAMUX request line channel (see the DMAREQ_ID field of the DMAMUX2_CyCR register and the DMAMUX mapping implementation section).

This register shall be written by an unprivileged or privileged write, according to the privileged mode of the considered DMAMUX request line multiplexer channel y it is assigned to, and considering that the DMAMUX request generator x channel output is selected by the y channel of the DMAMUX request line channel (see the DMAREQ_ID field of the DMAMUX2_CyCR register and the DMAMUX mapping implementation section).

Note: With the request generator channel x disabled (GE=0) and thus ready to be configured again, the software shall operate (at least) two separated write accesses to this register. First, set/update the number of DMA requests to be generated (bits GNBREQ[4:0]), enable the request generator (i.e. set bit 16 GE), and disable the detection of the generator trigger input (i.e. set GPOL[1:0] to 00). Next, enable the detection of the generator trigger input by selecting as rising/falling the edge to be detected (i.e. set GPOL[1:0] to 01, 10 or 11).

### Register Description

| Bit 31:24 | Reserved, must be kept at reset value. |
| Bit 23:19 | **GNBREQ[4:0]**: Number of DMA requests to be generated (minus 1) |
|           | Defines the number of DMA requests to be generated after a trigger event. The actual number of generated DMA requests is GNBREQ+1. |
|           | **Note:** This field shall only be written when GE bit is disabled. |
| Bit 18:17 | **GPOL[1:0]**: DMA request generator trigger polarity |
|           | Defines the edge polarity of the selected trigger input |
|           | 00: no event. I.e. none trigger detection nor generation. |
|           | 01: rising edge |
|           | 10: falling edge |
|           | 11: rising and falling edge |
| Bit 16    | **GE**: DMA request generator channel x enable |
|           | 0: DMA request generator channel x disabled |
|           | 1: DMA request generator channel x enabled |
| Bit 15:9  | Reserved, must be kept at reset value. |
| Bit 8     | **OIE**: Trigger overrun interrupt enable |
|           | 0: interrupt on a trigger overrun event occurrence is disabled |
|           | 1: interrupt on a trigger overrun event occurrence is enabled |
| Bit 7:5   | Reserved, must be kept at reset value. |
| Bit 4:0   | **SIG_ID[4:0]**: Signal identification |
|           | Selects the DMA request trigger input used for the channel x of the DMA request generator |
17.6.9 **DMAMUX1 request generator interrupt status register**

(DFAMUX1_RGSR)

Address offset: 0x140

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>OF7</td>
<td>OF6</td>
<td>OF5</td>
<td>OF4</td>
<td>OF3</td>
<td>OF2</td>
<td>OF1</td>
<td>OF0</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **OF[7:0]: Trigger overrun event flag**

The flag is set when a new trigger event occurs on DMA request generator channel x, before the request counter underrun (the internal request counter programmed via the GNBREQ field of the DMAMUX_RGxCR register).

The flag is cleared by writing 1 to the corresponding COFx bit in the DMAMUX_RGCFR register.

17.6.10 **DMAMUX2 request generator interrupt status register**

(DFAMUX2_RGSR)

Address offset: 0x140

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>OF7</td>
<td>OF6</td>
<td>OF5</td>
<td>OF4</td>
<td>OF3</td>
<td>OF2</td>
<td>OF1</td>
<td>OF0</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **OF[7:0]: Trigger overrun event flag**

The flag is set when a new trigger event occurs on DMA request generator channel x.

The flag is cleared by writing 1 to the corresponding COFx bit in the DMAMUX2_RGCFR register.
17.6.11 DMAMUX1 request generator interrupt clear flag register
(DMAMUX1_RGCFR)

Address offset: 0x144
Reset value: 0x0000 0000

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

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

Bits 7:0 COF[7:0]: Clear trigger overrun event flag
Writing 1 in each bit clears the corresponding overrun flag OFx in the DMAMUX_RGSR register.

17.6.12 DMAMUX2 request generator interrupt clear flag register
(DMAMUX2_RGCFR)

Address offset: 0x144
Reset value: 0x0000 0000

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

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

Bits 7:0 COF[7:0]: Clear trigger overrun event flag
Writing 1 in each bit clears the corresponding overrun flag OFx in the DMAMUX2_RGSR register.
17.6.13 DMAMUX register map

The following table summarizes the DMAMUX registers and reset values. Refer to the register boundary address table for the DMAMUX register base address.

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | DMAMUX_COCR(1)(2) | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x004  | DMAMUX_C1CR(1)(2) | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x008  | DMAMUX_C2CR(1)(2) | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x00C  | DMAMUX_C3CR(1)(2) | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x010  | DMAMUX_C4CR(1)(2) | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x014  | DMAMUX_C5CR(1)(2) | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x018  | DMAMUX_C6CR(1)(2) | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x01C  | DMAMUX_C7CR(1)(2) | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x020  | DMAMUX_C8CR(3)   | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x024  | DMAMUX_C9CR(4)   | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x028  | DMAMUX_C10CR(3)  | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x02C  | DMAMUX_C11CR(3)  | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x030  | DMAMUX_C12CR(3)  | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x034  | DMAMUX_C13CR(3)  | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x038  | DMAMUX_C14CR(3)  | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x03C  | DMAMUX_C15CR(3)  | SYNC_ID [2:0] | NBREQ[4:0] | OPE | ETE | EGE | DMAREQ_ID[6:0] |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |

| 0x040 - 0x07C | Reserved |

Table 109. DMAMUX register map and reset values
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 31-0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x080</td>
<td>DMAMUX_CSR</td>
<td>0x080</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x084</td>
<td>DMAMUX_CFR</td>
<td>0x084</td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x084-0x0FC</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x100</td>
<td>DMAMUX_RG0CR</td>
<td>0x100</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x104</td>
<td>DMAMUX_RG1CR</td>
<td>0x104</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x108</td>
<td>DMAMUX_RG2CR</td>
<td>0x108</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x10C</td>
<td>DMAMUX_RG3CR</td>
<td>0x10C</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x110</td>
<td>DMAMUX_RG4CR</td>
<td>0x110</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x114</td>
<td>DMAMUX_RG5CR</td>
<td>0x114</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x118</td>
<td>DMAMUX_RG6CR</td>
<td>0x118</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x11C</td>
<td>DMAMUX_RG7CR</td>
<td>0x11C</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x140</td>
<td>DMAMUX_RGSR</td>
<td>0x140</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x144</td>
<td>DMAMUX_RGCFR</td>
<td>0x144</td>
<td>Reset value</td>
</tr>
</tbody>
</table>

1. For DMAMUX2 bits 6:5 are reserved.
2. For DMAMUX2 bits 27:24 correspond to SYNC_ID[3:0]
3. Only applies to DMAMUX1. For DMAMUX2 the word is reserved.
4. For DMAMUX2 bits 15:8 are reserved.
5. Valid for DMAMUX1. For DMAMUX2 bits 4:0 are for SIG_ID[4:0].

Refer to Section 2.3 on page 128 for the register boundary addresses.
18 Chrom-Art Accelerator™ controller (DMA2D)

18.1 DMA2D introduction

The Chrom-Art Accelerator™ (DMA2D) is a specialized DMA dedicated to image manipulation. It can perform the following operations:

- Filling a part or the whole of a destination image with a specific color
- Copying a part or the whole of a source image into a part or the whole of a destination image
- Copying a part or the whole of a source image into a part or the whole of a destination image with a pixel format conversion
- Blending a part and/or two complete source images with different pixel format and copy the result into a part or the whole of a destination image with a different color format.

All the classical color coding schemes are supported from 4-bit up to 32-bit per pixel with indexed or direct color mode, including block based YCbCr to handle JPEG decoder output. The DMA2D has its own dedicated memories for CLUTs (color look-up tables).

18.2 DMA2D main features

The main DMA2D features are:

- A single AXI master bus architecture.
- AHB slave programming interface supporting 8/16/32-bit accesses (except for CLUT accesses which are 32-bit).
- User programmable working area size
- User programmable offset for sources and destination areas
- User programmable sources and destination addresses on the whole memory space
- Up to 2 sources with blending operation
- Alpha value can be modified (source value, fixed value or modulated value)
- User programmable source and destination color format
- Up to 11 color formats supported from 4-bit up to 32-bit per pixel with indirect or direct color coding
- Block based (8x8) YCbCr support with 4:4:4, 4:2:2 and 4:2:0 chroma sub-sampling factors
- 2 internal memories for CLUT storage in indirect color mode
- Automatic CLUT loading or CLUT programming via the CPU
- User programmable CLUT size
- Internal timer to control AXI bandwidth
- 6 operating modes: register-to-memory, memory-to-memory, memory-to-memory with pixel format conversion, memory-to-memory with pixel format conversion and blending, memory-to memory with pixel format conversion, blending and fixed color foreground.
and memory-to memory with pixel format conversion, blending and fixed color background.

- Area filling with a fixed color
- Copy from an area to another
- Copy with pixel format conversion between source and destination images
- Copy from two sources with independent color format and blending
- Output buffer byte swapping to support refresh of displays through parallel interface
- Abort and suspend of DMA2D operations
- Watermark interrupt on a user programmable destination line
- Interrupt generation on bus error or access conflict
- Interrupt generation on process completion

18.3 DMA2D functional description

18.3.1 General description

The DMA2D controller performs direct memory transfer. As an AXI master, it can take the control of the AXI bus matrix to initiate AXI transactions.

The DMA2D can operate in the following modes:

- Register-to-memory
- Memory-to-memory
- Memory-to-memory with Pixel Format Conversion
- Memory-to-memory with Pixel Format Conversion and Blending
- Memory-to memory with pixel format conversion, blending and fixed color foreground

The AHB slave port is used to program the DMA2D controller.

The block diagram of the DMA2D is shown in Figure 90: DMA2D block diagram.
18.4 DMA2D pins and internal signals

*Table 110* lists the DMA2D internal signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dma2d_aclk</td>
<td>Input</td>
<td>32-bit AXI bus clock</td>
</tr>
<tr>
<td>dma2d_gbl_it</td>
<td>Output</td>
<td>DMA2D global interrupt</td>
</tr>
<tr>
<td>dma2d_clut_trg</td>
<td>Output</td>
<td>CLUT transfer complete (to MDMA)</td>
</tr>
<tr>
<td>dma2d_tc_trg</td>
<td>Output</td>
<td>Transfer complete (to MDMA)</td>
</tr>
<tr>
<td>dma2d_tw_trg</td>
<td>Output</td>
<td>Transfer watermark (to MDMA)</td>
</tr>
</tbody>
</table>
18.4.1 DMA2D control

The DMA2D controller is configured through the DMA2D Control Register (DMA2D_CR) which allows selecting:

The user application can perform the following operations:
- Select the operating mode
- Enable/disable the DMA2D interrupt
- Start/suspend/abort ongoing data transfers

18.4.2 DMA2D foreground and background FIFOs

The DMA2D foreground (FG) FG FIFO and background (BG) FIFO fetch the input data to be copied and/or processed.

The FIFOs fetch the pixels according to the color format defined in their respective pixel format converter (PFC).

They are programmed through a set of control registers:
- DMA2D foreground memory address register (DMA2D_FGMAR)
- DMA2D foreground offset register (DMA2D_FGOR)
- DMA2D background memory address register (DMA2D_BGMAR)
- DMA2D background offset register (DMA2D_BGBOR)
- DMA2D number of lines register (number of lines and pixel per lines) (DMA2D_NLR)

When the DMA2D operates in register-to-memory mode, none of the FIFOs is activated.

When the DMA2D operates in memory-to-memory mode (no pixel format conversion nor blending operation), only the FG FIFO is activated and acts as a buffer.

When the DMA2D operates in memory-to-memory operation with pixel format conversion (no blending operation), the BG FIFO is not activated.

18.4.3 DMA2D foreground and background pixel format converter (PFC)

DMA2D foreground pixel format converter (PFC) and background pixel format converter perform the pixel format conversion to generate a 32-bit per pixel value. The PFC can also modify the alpha channel.

The first stage of the converter converts the color format. The original color format of the foreground pixel and background pixels are configured through the CM[3:0] bits of the DMA2D_FGPFCCR and DMA2D_BGPFCCR, respectively.

The supported input formats are given in Table 111: Supported color mode in input.

<table>
<thead>
<tr>
<th>CM[3:0]</th>
<th>Color mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>ARGB8888</td>
</tr>
<tr>
<td>0001</td>
<td>RGB888</td>
</tr>
<tr>
<td>0010</td>
<td>RGB565</td>
</tr>
<tr>
<td>0011</td>
<td>ARGB1555</td>
</tr>
<tr>
<td>0100</td>
<td>ARGB4444</td>
</tr>
</tbody>
</table>
The color format are coded as follows:

- Alpha value field: transparency
  - 0xFF value corresponds to an opaque pixel and 0x00 to a transparent one.
- R field for Red
- G field for Green
- B field for Blue
- L field: luminance
  - This field is the index to a CLUT to retrieve the three/four RGB/ARGB components.

If the original format was direct color mode, then the extension to 8-bit per channel is performed by copying the MSBs into the LSBs. This ensures a perfect linearity of the conversion.

If the original format does not include an alpha channel, the alpha value is automatically set to 0xFF (opaque).

If the original format is indirect color mode, a CLUT is required and each pixel format converter is associated with a 256 entry 32-bit CLUT.

For the specific alpha mode A4 and A8, no color information is stored nor indexed. The color to be used for the image generation is fixed and is defined in the DMA2D_FGCOLOR for foreground pixels and in the DMA2D_BGCOLOR register for background pixels.

The order of the fields in the system memory is defined in Table 112: Data order in memory.

### Table 111: Supported color mode in input

<table>
<thead>
<tr>
<th>CM[3:0]</th>
<th>Color mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0101</td>
<td>L8</td>
</tr>
<tr>
<td>0110</td>
<td>AL44</td>
</tr>
<tr>
<td>0111</td>
<td>AL88</td>
</tr>
<tr>
<td>1000</td>
<td>L4</td>
</tr>
<tr>
<td>1001</td>
<td>A8</td>
</tr>
<tr>
<td>1010</td>
<td>A4</td>
</tr>
<tr>
<td>1011</td>
<td>YCbCr (only for foreground)</td>
</tr>
</tbody>
</table>

For the specific alpha mode A4 and A8, no color information is stored nor indexed. The color to be used for the image generation is fixed and is defined in the DMA2D_FGCOLOR for foreground pixels and in the DMA2D_BGCOLOR register for background pixels.

The order of the fields in the system memory is defined in Table 112: Data order in memory.
The 24-bit RGB888 aligned on 32-bit is supported through the ARGB8888 mode.

Once the 32-bit value is generated, the alpha channel can be modified according to the \( \text{AM}[1:0] \) field of the DMA2D_FGPFCCR/DMA2D_BGPFCCR registers as shown in Table 113: Alpha mode configuration.

The alpha channel can be:

- kept as it is (no modification),
- replaced by the \( \text{ALPHA}[7:0] \) value of DMA2D_FGPFCCR/DMA2D_BGPFCCR,
- or replaced by the original alpha value multiplied by the \( \text{ALPHA}[7:0] \) value of DMA2D_FGPFCCR/DMA2D_BGPFCCR divided by 255.

### Table 112. Data order in memory

<table>
<thead>
<tr>
<th>Color Mode</th>
<th>@ + 3</th>
<th>@ + 2</th>
<th>@ + 1</th>
<th>@ + 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARGB8888</td>
<td>( A_0[7:0] )</td>
<td>( R_0[7:0] )</td>
<td>( G_0[7:0] )</td>
<td>( B_0[7:0] )</td>
</tr>
<tr>
<td>RGB888</td>
<td>( B_1[7:0] )</td>
<td>( R_0[7:0] )</td>
<td>( G_0[7:0] )</td>
<td>( B_0[7:0] )</td>
</tr>
<tr>
<td></td>
<td>( G_2[7:0] )</td>
<td>( B_2[7:0] )</td>
<td>( R_1[7:0] )</td>
<td>( G_1[7:0] )</td>
</tr>
<tr>
<td></td>
<td>( R_3[7:0] )</td>
<td>( G_3[7:0] )</td>
<td>( B_3[7:0] )</td>
<td>( R_2[7:0] )</td>
</tr>
<tr>
<td>L8</td>
<td>( L_3[7:0] )</td>
<td>( L_2[7:0] )</td>
<td>( L_1[7:0] )</td>
<td>( L_0[7:0] )</td>
</tr>
<tr>
<td>AL88</td>
<td>( A_1[7:0] )</td>
<td>( L_1[7:0] )</td>
<td>( A_0[7:0] )</td>
<td>( L_0[7:0] )</td>
</tr>
<tr>
<td>A8</td>
<td>( A_3[7:0] )</td>
<td>( A_2[7:0] )</td>
<td>( A_1[7:0] )</td>
<td>( A_0[7:0] )</td>
</tr>
</tbody>
</table>

### Table 113. Alpha mode configuration

<table>
<thead>
<tr>
<th>AM[1:0]</th>
<th>Alpha mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>No modification</td>
</tr>
<tr>
<td>01</td>
<td>Replaced by value in DMA2D_xxPFCCR</td>
</tr>
<tr>
<td>10</td>
<td>Replaced by original value multiplied by the value in DMA2D_xxPFCCR / 255</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Note: To support the alternate format, the incoming alpha value can be inverted setting the AI bit of the DMA2D_FGPFCCR/DMA2D_BGPFCCR registers. This applies also to the Alpha value stored in the DMA2D_FGPFCCR/DMA2D_BGPFCCR and in the CLUT.

The R and B fields can also be swapped setting the RBS bit of the DMA2D_FGPFCCR/DMA2D_BGPFCCR registers. This applies also to the RGB order used in the CLUT and in the DMA2D_FGCOLR/DMA2D_BGCOLR registers.
18.4.4 DMA2D foreground and background CLUT interface

The CLUT interface manages the CLUT memory access and the automatic loading of the CLUT.

Three kinds of accesses are possible:

- CLUT read by the PFC during pixel format conversion operation
- CLUT accessed through the AHB slave port when the CPU is reading or writing data into the CLUT
- CLUT written through the AXI master port when an automatic loading of the CLUT is performed

The CLUT memory loading can be done in two different ways:

- Automatic loading
  The following sequence should be followed to load the CLUT:
  a) Program the CLUT address into the DMA2D_FGCMAR register (foreground CLUT) or DMA2D_BGCMAR register (background CLUT)
  b) Program the CLUT size in the CS[7:0] field of the DMA2D_FGPFCCR register (foreground CLUT) or DMA2D_BGPFCCR register (background CLUT).
  c) Set the START bit of the DMA2D_FGPFCCR register (foreground CLUT) or DMA2D_BGPFCCR register (background CLUT) to start the transfer. During this automatic loading process, the CLUT is not accessible by the CPU. If a conflict occurs, a CLUT access error interrupt is raised assuming CAEIE is set to ‘1’ in DMA2D_CR.

- Manual loading
  The application has to program the CLUT manually through the DMA2D AHB slave port to which the local CLUT memory is mapped. The foreground CLUT is located at address offset 0x0400 and the background CLUT at address offset 0x0800.

The CLUT format can be 24 or 32 bits. It is configured through the CCM bit of the DMA2D_FGPFCCR register (foreground CLUT) or DMA2D_BGPFCCR register (background CLUT) as shown in Table 114: Supported CLUT color mode.

Table 114. Supported CLUT color mode

<table>
<thead>
<tr>
<th>CCM</th>
<th>CLUT color mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32-bit ARGB8888</td>
</tr>
<tr>
<td>1</td>
<td>24-bit RGB8888</td>
</tr>
</tbody>
</table>

The way the CLUT data are organized in the system memory is specified in Table 115: CLUT data order in memory.

Table 115. CLUT data order in memory

<table>
<thead>
<tr>
<th>CLUT Color Mode</th>
<th>@ + 3</th>
<th>@ + 2</th>
<th>@ + 1</th>
<th>@ + 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARGB8888</td>
<td>A0[7:0]</td>
<td>R0[7:0]</td>
<td>G0[7:0]</td>
<td>B0[7:0]</td>
</tr>
<tr>
<td>RGB8888</td>
<td>B1[7:0]</td>
<td>R0[7:0]</td>
<td>G0[7:0]</td>
<td>B0[7:0]</td>
</tr>
<tr>
<td></td>
<td>R3[7:0]</td>
<td>G3[7:0]</td>
<td>B3[7:0]</td>
<td>R2[7:0]</td>
</tr>
</tbody>
</table>
18.4.5 DMA2D blender

The DMA2D blender blends the source pixels by pair to compute the resulting pixel. The blending is performed according to the following equation:

\[
\begin{align*}
\alpha_{\text{Mult}} &= \frac{\alpha_{\text{FG}} \cdot \alpha_{\text{BG}}}{255} \\
\alpha_{\text{OUT}} &= \alpha_{\text{FG}} + \alpha_{\text{BG}} - \alpha_{\text{Mult}} \\
C_{\text{OUT}} &= \frac{C_{\text{FG}} \cdot \alpha_{\text{FG}} + C_{\text{BG}} \cdot \alpha_{\text{BG}} - C_{\text{BG}} \cdot \alpha_{\text{Mult}}}{\alpha_{\text{OUT}}} \quad \text{with } C = \text{R or G or B}
\end{align*}
\]

Division is rounded to the nearest lower integer

No configuration register is required by the blender. The blender usage depends on the DMA2D operating mode defined in MODE[1:0] field of the DMA2D_CR register.

18.4.6 DMA2D output PFC

The output PFC performs the pixel format conversion from 32 bits to the output format defined in the CM[2:0] field of the DMA2D output pixel format converter configuration register (DMA2D_OPFCCR).

The supported output formats are given in Table 116: Supported color mode in output.

<table>
<thead>
<tr>
<th>CM[2:0]</th>
<th>Color mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>ARGB8888</td>
</tr>
<tr>
<td>001</td>
<td>RGB888</td>
</tr>
<tr>
<td>010</td>
<td>RGB565</td>
</tr>
<tr>
<td>011</td>
<td>ARGB1555</td>
</tr>
<tr>
<td>100</td>
<td>ARGB4444</td>
</tr>
</tbody>
</table>

Note: To support the alternate format, the calculated alpha value can be inverted setting the AI bit of the DMA2D_OPFCCR registers. This applies also to the Alpha value used in the DMA2D_OCOLOR.

The R and B fields can also be swapped setting the RBS bit of the DMA2D_OPFCCR registers. This applies also to the RGB order used in the DMA2D_OCOLOR.
18.4.7 DMA2D output FIFO

The output FIFO programs the pixels according to the color format defined in the output PFC.

The destination area is defined through a set of control registers:
- DMA2D output memory address register (DMA2D_OMAR)
- DMA2D output offset register (DMA2D_OOR)
- DMA2D number of lines register (number of lines and pixel per lines) (DMA2D_NLR)

If the DMA2D operates in register-to-memory mode, the configured output rectangle is filled by the color specified in the DMA2D output color register (DMA2D_OCOLR) which contains a fixed 32-bit, 24-bit or 16-bit value. The format is selected by the CM[2:0] field of the DMA2D_OPFCCR register.

The data are stored into the memory in the order defined in Table 117: Data order in memory.

Table 117. Data order in memory

<table>
<thead>
<tr>
<th>Color mode</th>
<th>@ + 3</th>
<th>@ + 2</th>
<th>@ + 1</th>
<th>@ + 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARGB8888</td>
<td>A0[7:0]</td>
<td>R0[7:0]</td>
<td>G0[7:0]</td>
<td>B0[7:0]</td>
</tr>
<tr>
<td>RGB888</td>
<td>B1[7:0]</td>
<td>R0[7:0]</td>
<td>G0[7:0]</td>
<td>B0[7:0]</td>
</tr>
<tr>
<td></td>
<td>R3[7:0]</td>
<td>G3[7:0]</td>
<td>B3[7:0]</td>
<td>R2[7:0]</td>
</tr>
</tbody>
</table>

The RGB888 aligned on 32-bit is supported through the ARGB8888 mode.

18.4.8 DMA2D output FIFO byte reordering

The output FIFO bytes can be reordered to support display frame buffer update through a parallel interface (F(S)MC) directly from the DMA2D.

The reordering of bytes can be done using:
- RBS bit to swap red and blue component
- SB bit to swap byte two by two in the output FIFO

When the byte swapping is activated (SB field of the DMA2D_OPFCR is set), the number of pixel per line (PL field of the DMA2D_NLR) must be even and the output memory address (MA field of the DMA2D_OMAR) must be even, and the output line offset computed in bytes (resulting from LOM field of DMA2D_CR and LO field of DMA2D_OOR values) must be even. If not a configuration error will be detected.
16-bit mode (RGB565)
This mode is supported without byte reordering by the DMA2D.

Figure 91. Intel 8080 16-bit mode (RGB565)

18/24-bit mode (RGB888)
This mode needs data reordering.
1. The red and the blue have to be swapped (setting the RBS bit)
2. The MSB and the LSB bytes of an half-word as to be swapped (setting the SB bit)

Figure 92. Intel 8080 18/24-bit mode (RGB888)
An 8-bit timer is embedded into the AXI master port to provide an optional limitation of the bandwidth on the crossbar.

This timer is clocked by the AXI clock and counts a dead time between two consecutive accesses. This limits the bandwidth usage.

The timer enabling and the dead time value are configured through the AXI master port timer configuration register (DMA2D_AMPTCR).

### DMA2D AXI master port timer

### DMA2D transactions

DMA2D transactions consist of a sequence of a given number of data transfers. The number of data and the width can be programmed by software.

Each DMA2D data transfer is composed of up to 4 steps:

1. Data loading from the memory location pointed by the DMA2D_FGMAR register and pixel format conversion as defined in DMA2D_FGCR.
2. Data loading from a memory location pointed by the DMA2D_BGMAR register and pixel format conversion as defined in DMA2D_BGCR.
3. Blending of all retrieved pixels according to the alpha channels resulting of the PFC operation on alpha values.
4. Pixel format conversion of the resulting pixels according to the DMA2D_OCR register and programming of the data to the memory location addressed through the DMA2D_OMAR register.
18.4.11 DMA2D configuration

Both source and destination data transfers can target peripherals and memories in the whole 4-GB memory area, at addresses ranging between 0x0000 0000 and 0xFFFF FFFF.

The DMA2D can operate in any of the four following modes selected through $\text{MODE}[1:0]$ bits of the DMA2D\_CR register:

- Register-to-memory
- Memory-to-memory
- Memory-to-memory with PFC
- Memory-to-memory with PFC and blending
- Memory-to-memory with PFC, blending and fixed FG color

Register-to-memory

The register-to-memory mode is used to fill a user defined area with a predefined color. The color format is set in the DMA2D\_OPFCCR.

The DMA2D does not perform any data fetching from any source. It just writes the color defined in the DMA2D\_OCOLR register to the area located at the address pointed by the DMA2D\_OMAR and defined in the DMA2D\_NLR and DMA2D\_OOR.

Memory-to-memory

In memory-to-memory mode, the DMA2D does not perform any graphical data transformation. The foreground input FIFO acts as a buffer and the data are transferred from the source memory location defined in DMA2D\_FGMAR to the destination memory location pointed by DMA2D\_OMAR.

The color mode programmed in the CM[3:0] bits of the DMA2D\_FGPFCCR register defines the number of bits per pixel for both input and output.

The size of the area to be transferred is defined by the DMA2D\_NLR and DMA2D\_FGOR registers for the source, and by DMA2D\_NLR and DMA2D\_OOR registers for the destination.

Memory-to-memory with PFC

In this mode, the DMA2D performs a pixel format conversion of the source data and stores them in the destination memory location.

The size of the areas to be transferred are defined by the DMA2D\_NLR and DMA2D\_FGOR registers for the source, and by DMA2D\_NLR and DMA2D\_OOR registers for the destination.

Data are fetched from the location defined in the DMA2D\_FGMAR register and processed by the foreground PFC. The original pixel format is configured through the DMA2D\_FGPFCCR register.

If the original pixel format is direct color mode, then the color channels are all expanded to 8 bits.

If the pixel format is indirect color mode, the associated CLUT has to be loaded into the CLUT memory.

The CLUT loading can be done automatically by following the sequence below:
1. Set the CLUT address into the DMA2D_FGCMAR.
2. Set the CLUT size in the CS[7:0] bits of the DMA2D_FGPFCCR register.
3. Set the CLUT format (24 or 32 bits) in the CCM bit of the DMA2D_FGPFCCR register.
4. Start the CLUT loading by setting the START bit of the DMA2D_FGPFCCR register.

Once the CLUT loading is complete, the CTCIF flag of the DMA2D_IFR register is raised, and an interrupt is generated if the CTCIE bit is set in DMA2D_CR. The automatic CLUT loading process cannot work in parallel with classical DMA2D transfers.

The CLUT can also be filled by the CPU or by any other master through the AHB port. The access to the CLUT is not possible when a DMA2D transfer is ongoing and uses the CLUT (indirect color format).

In parallel to the color conversion process, the alpha value can be added or changed depending on the value programmed in the DMA2D_FGPFCCR register. If the original image does not have an alpha channel, a default alpha value of 0xFF is automatically added to obtain a fully opaque pixel. The alpha value can be modified according to the AM[1:0] bits of the DMA2D_FGPFCCR register:

- It can be unchanged.
- It can be replaced by the value defined in the ALPHA[7:0] value of the DMA2D_FGPFCCR register.
- It can be replaced by the original value multiplied by the ALPHA[7:0] value of the DMA2D_FGPFCCR register divided by 255.

The resulting 32-bit data are encoded by the OUT PFC into the format specified by the CM[2:0] field of the DMA2D_OPFCCR register. The output pixel format cannot be the indirect mode since no CLUT generation process is supported.

The processed data are written into the destination memory location pointed by DMA2D_OMAR.

**Memory-to-memory with PFC and blending**

In this mode, 2 sources are fetched in the foreground FIFO and background FIFO from the memory locations defined by DMA2D_FGMAR and DMA2D_BGMAR.

The two pixel format converters have to be configured as described in the memory-to-memory mode. Their configurations can be different as each pixel format converter is independent and have their own CLUT memory.

Once each pixel has been converted into 32 bits by their respective PFCs, they are blended according to the equation below:

$$\alpha_{OUT} = \alpha_{FG} + \alpha_{BG} - \alpha_{Mult}$$

$$C_{OUT} = \frac{C_{FG} \alpha_{FG} + C_{BG} \alpha_{BG} - C_{BG} \alpha_{Mult}}{\alpha_{OUT}}$$

with $C = R$ or $G$ or $B$

*Division are rounded to the nearest lower integer*
The resulting 32-bit pixel value is encoded by the output PFC according to the specified output format, and the data are written into the destination memory location pointed by DMA2D_OMAR.

**Memory-to-memory with PFC, blending and fixed color FG**

In this mode, only 1 source is fetched in the background FIFO from the memory location defined by DMA2D_BGMAR.

The value of the foreground color is given by the DMA2D_FGCOLOR register and the alpha value is set to 0xFF (opaque).

The alpha value can be replaced or modified according to the AM[1:0] and ALPHA[7:0] fields of the DMA2D_FGPFCR.

The two pixel format converters have to be configured as described in the memory-to-memory mode. Their configurations can be different as each pixel format converter are independent and have their own CLUT memory.

Once each pixel has been converted into 32 bits by their respective PFCs, they are blended together, and the resulting 32-bit pixel value is encoded by the output PFC according to the specified output format, and the data are written into the destination memory location pointed by DMA2D_OMAR.

**Memory-to-memory with PFC, blending and fixed color BG**

In this mode, only 1 source is fetched in the foreground FIFO from the memory location defined by DMA2D_FGMAR.

The value of the background color is given by the DMA2D_BGCOLOR register and the alpha value is set to 0xFF (opaque).

The alpha value can be replaced or modified according to the AM[1:0] and ALPHA[7:0] fields of the DMA2D_BGPFCR.

The two pixel format converters have to be configured as described in the memory-to-memory mode. Their configurations can be different as each pixel format converter are independent and have their own CLUT memory.

Once each pixel has been converted into 32 bits by their respective PFCs, they are blended together, and the resulting 32-bit pixel value is encoded by the output PFC according to the specified output format, and the data are written into the destination memory location pointed by DMA2D_OMAR.

**Configuration error detection**

The DMA2D checks that the configuration is correct before any transfer. The configuration error interrupt flag is set by hardware when a wrong configuration is detected when a new transfer/automatic loading starts. An interrupt is then generated if the CEIE bit of the DMA2D_CR is set.
The wrong configurations that can be detected are listed below:

- Foreground CLUT automatic loading: MA bits of DMA2D_FGCMAR are not aligned with CCM of DMA2D_FGPFCCR.
- Background CLUT automatic loading: MA bits of DMA2D_BGCMAR are not aligned with CCM of DMA2D_BGPFCCR.
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): MA bits of DMA2D_FGMAR are not aligned with CM of DMA2D_FGPFCCR.
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): CM bits of DMA2D_FGPFCCR are invalid.
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): PL bits of DMA2D_NLR are odd while CM of DMA2D_FGPFCCR is A4 or L4.
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): LO bits of DMA2D_FGOR are odd while CM of DMA2D_FGPFCCR is A4 or L4 and LOM bit of the DMA2D_CR is pixel mode.
- Memory transfer (only in blending mode and except in memory-to-memory mode with blending and fixed color FG): MA bits of DMA2D_BGMAR are not aligned with the CM of DMA2D_BGPFCCR.
- Memory transfer: (only in blending mode and in blending with fixed color FG mode) CM bits of DMA2D_BGPFCCR are invalid.
- Memory transfer: (only in blending mode and in blending with fixed color FG mode): PL bits of DMA2D_NLR odd while CM of DMA2D_BGPFCCR is A4 or L4.
- Memory transfer (only in blending mode and in blending with fixed color FG mode): LO bits of DMA2D_BGOR are odd while CM of DMA2D_BGPFCCR is A4 or L4 and LOM bit of the DMA2D_CR is pixel mode.
- Memory transfer (except in memory to memory mode): MA bits of DMA2D_OMAR are not aligned with CM bits of DMA2D_OPFCCR.
- Memory transfer (except in memory to memory mode): CM bits of DMA2D_OPFCCR are invalid.
- Memory transfer with byte swapping: PL bits of DMA2D_NLR are odd or MA bits of the DMA2D_OMAR are odd or LO in bytes (resulting from LOM bit of the DMA2D_CR and LO bits of DMA2D_OOR values) are odd while SB bit of DMA2D_OPFCCR is set.
- Memory transfer: NL bits of DMA2D_NLR = 0.
- Memory transfer: PL bits of DMA2D_NLR = 0.
- Memory transfer: MODE bits of DMA2D_CR are invalid.
- YCbCr format: when a CLUT loading starts setting the START bit of the DMA2D_FGPFCCR.
- YCbCr format: when the memory-to-memory mode is selected.
- YCbCr format: when YCbCr4:4:4 is selected and the sum of the number of pixel (PL) and the line offset LO is not a multiple of 8 pixels.
- YCbCr format: when YCbCr4:2:2 or YCbCr4:2:0 is selected and the sum of the number of pixel (PL) and the line offset LO is not a multiple of 16 pixels.
18.4.12 **YCbCr support**

The DMA2D foreground plane can support 8x8 block based YCbCr as output by the JPEG decoder with different chroma sub-sampling factors:

The memory organization follows the standard JFIF rules:
- Each of the three color component must be coded on 8-bit
- Each component must be arranged by blocks of 8x8 (64 bytes) called MCU

Depending of the chroma sub-sampling factor the MCU must be arranged in the memory as described in *Table 120: MCU order in memory*.

**Table 120. MCU order in memory**

<table>
<thead>
<tr>
<th>Sub-sampling</th>
<th>@</th>
<th>@ + 64</th>
<th>@ + 128</th>
<th>@+192</th>
<th>@+256</th>
<th>@ + 320</th>
</tr>
</thead>
<tbody>
<tr>
<td>4:4:4</td>
<td>Y₁</td>
<td>Cb₁</td>
<td>Cr₁</td>
<td>Y₂</td>
<td>Cb₂</td>
<td>Cr₂</td>
</tr>
<tr>
<td>4:2:2</td>
<td>Y₁</td>
<td>Y₂</td>
<td>Cb₁₂</td>
<td>Cr₁₂</td>
<td>Y₃</td>
<td>Y₄</td>
</tr>
<tr>
<td>4:2:0</td>
<td>Y₁</td>
<td>Y₂</td>
<td>Y₃</td>
<td>Y₄</td>
<td>Cb₁₂₃₄</td>
<td>Cr₁₂₃₄</td>
</tr>
</tbody>
</table>

The chroma sub-sampling factor is configured through the CSS field of the DMA2D_FGPFCCR register.

Once the DMA2D has started with the foreground configured in YCbCr color mode, the first 2 chroma MCU are loaded in the foreground CLUT. Once the chroma MCU are loaded, the DMA2D performs the loading of the Y MCU as for a classical color mode.

18.4.13 **DMA2D transfer control (start, suspend, abort and completion)**

Once the DMA2D is configured, the transfer can be launched by setting the START bit of the DMA2D_CR register. Once the transfer is completed, the START bit is automatically reset and the TCIF flag of the DMA2D_ISR register is raised. An interrupt can be generated if the TCIE bit of the DMA2D_CR is set.

The user application can suspend the DMA2D at any time by setting the SUSP bit of the DMA2D_CR register. The transaction can then be aborted by setting the ABORT bit of the DMA2D_CR register or can be restarted by resetting the SUSP bit of the DMA2D_CR register.

The user application can abort at any time an ongoing transaction by setting the ABORT bit of the DMA2D_CR register. In this case, the TCIF flag is not raised.

Automatic CLUT transfers can also be aborted or suspended by using their own START bits in the DMA2D_FGPFCCR and DMA2D_BGPFCCR registers.

18.4.14 **Watermark**

A watermark can be programmed to generate an interrupt when the last pixel of a given line has been written to the destination memory area.

The line number is defined in the LW[15:0] field of the DMA2D_LWR register.

When the last pixel of this line has been transferred, the TWIF flag of the DMA2D_ISR register is raised and an interrupt is generated if the TWIE bit of the DMA2D_CR is set.
18.4.15 Error management

Two kind of errors can be triggered:
- AXI master port errors signaled by the TEIF flag of the DMA2D_ISR register.
- Conflicts caused by CLUT access (CPU trying to access the CLUT while a CLUT loading or a DMA2D transfer is ongoing) signaled by the CAEIF flag of the DMA2D_ISR register.

Both flags are associated to their own interrupt enable flag in the DMA2D_CR register to generate an interrupt if need be (TEIE and CAEIE).

18.4.16 AXI dead time

To limit the AXI bandwidth usage, a dead time between two consecutive AXI accesses can be programmed.

This feature can be enabled by setting the EN bit in the DMA2D_AMTCR register.

The dead time value is stored in the DT[7:0] field of the DMA2D_AMTCR register. This value represents the guaranteed minimum number of cycles between two consecutive transactions on the AXI bus.

The update of the dead time value while the DMA2D is running will be taken into account for the next AXI transfer.

18.5 DMA2D interrupts

An interrupt can be generated on the following events:
- Configuration error
- CLUT transfer complete
- CLUT access error
- Transfer watermark reached
- Transfer complete
- Transfer error

Separate interrupt enable bits are available for flexibility.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration error</td>
<td>CEIF</td>
<td>CEIE</td>
</tr>
<tr>
<td>CLUT transfer complete</td>
<td>CTCIF</td>
<td>CTCIE</td>
</tr>
<tr>
<td>CLUT access error</td>
<td>CAEIF</td>
<td>CAEIE</td>
</tr>
<tr>
<td>Transfer watermark</td>
<td>TWF</td>
<td>TWIE</td>
</tr>
<tr>
<td>Transfer complete</td>
<td>TCIF</td>
<td>TCIE</td>
</tr>
<tr>
<td>Transfer error</td>
<td>TEIF</td>
<td>TEIE</td>
</tr>
</tbody>
</table>
18.6 DMA2D registers

18.6.1 DMA2D control register (DMA2D_CR)

Address offset: 0x0000
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-19</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 18-16</td>
<td><strong>MODE [2:0]</strong>: DMA2D mode</td>
</tr>
<tr>
<td>This bit is set and cleared by software. It cannot be modified while a transfer is ongoing.</td>
<td></td>
</tr>
<tr>
<td>000: Memory-to-memory (FG fetch only)</td>
<td></td>
</tr>
<tr>
<td>001: Memory-to-memory with PFC (FG fetch only with FG PFC active)</td>
<td></td>
</tr>
<tr>
<td>010: Memory-to-memory with blending (FG and BG fetch with PFC and blending)</td>
<td></td>
</tr>
<tr>
<td>011: Register-to-memory (no FG nor BG, only output stage active)</td>
<td></td>
</tr>
<tr>
<td>100: Memory-to-memory with Blending and fixed color FG (BG fetch only with FG and BG PFC active)</td>
<td></td>
</tr>
<tr>
<td>101: Memory-to-memory with Blending and fixed color BG (BG fetch only with FG and BG PFC active)</td>
<td></td>
</tr>
<tr>
<td>others: meaningless</td>
<td></td>
</tr>
<tr>
<td>Bits 15-14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 13</td>
<td><strong>CEIE</strong>: Configuration Error Interrupt Enable</td>
</tr>
<tr>
<td>This bit is set and cleared by software.</td>
<td></td>
</tr>
<tr>
<td>0: CE interrupt disable</td>
<td></td>
</tr>
<tr>
<td>1: CE interrupt enable</td>
<td></td>
</tr>
<tr>
<td>Bit 12</td>
<td><strong>CTCIE</strong>: CLUT transfer complete interrupt enable</td>
</tr>
<tr>
<td>This bit is set and cleared by software.</td>
<td></td>
</tr>
<tr>
<td>0: CTC interrupt disable</td>
<td></td>
</tr>
<tr>
<td>1: CTC interrupt enable</td>
<td></td>
</tr>
<tr>
<td>Bit 11</td>
<td><strong>CAEIE</strong>: CLUT access error interrupt enable</td>
</tr>
<tr>
<td>This bit is set and cleared by software.</td>
<td></td>
</tr>
<tr>
<td>0: CAE interrupt disable</td>
<td></td>
</tr>
<tr>
<td>1: CAE interrupt enable</td>
<td></td>
</tr>
<tr>
<td>Bit 10</td>
<td><strong>TWIE</strong>: Transfer watermark interrupt enable</td>
</tr>
<tr>
<td>This bit is set and cleared by software.</td>
<td></td>
</tr>
<tr>
<td>0: TW interrupt disable</td>
<td></td>
</tr>
<tr>
<td>1: TW interrupt enable</td>
<td></td>
</tr>
</tbody>
</table>
Bit 9 **TCIE**: Transfer complete interrupt enable
   This bit is set and cleared by software.
   0: TC interrupt disable
   1: TC interrupt enable

Bit 8 **TEIE**: Transfer error interrupt enable
   This bit is set and cleared by software.
   0: TE interrupt disable
   1: TE interrupt enable

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LOM**: Line Offset Mode
   This bit configures how is expressed the line offset (pixels or bytes) for the foreground, background and output.
   This bit is set and cleared by software. It can not be modified while a transfer is ongoing.
   0: Line offsets are expressed in pixels
   1: Line offsets are expressed in bytes

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **ABORT**: Abort
   This bit can be used to abort the current transfer. This bit is set by software and is automatically reset by hardware when the START bit is reset.
   0: No transfer abort requested
   1: Transfer abort requested

Bit 1 **SUSP**: Suspend
   This bit can be used to suspend the current transfer. This bit is set and reset by software. It is automatically reset by hardware when the START bit is reset.
   0: Transfer not suspended
   1: Transfer suspended

Bit 0 **START**: Start
   This bit can be used to launch the DMA2D according to the parameters loaded in the various configuration registers. This bit is automatically reset by the following events:
   - At the end of the transfer
   - When the data transfer is aborted by the user application by setting the ABORT bit in DMA2D_CR
   - When a data transfer error occurs
   - When the data transfer has not started due to a configuration error or another transfer operation already ongoing (automatic CLUT loading).
18.6.2 DMA2D interrupt status register (DMA2D_ISR)

Address offset: 0x0004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>CEIF: Configuration error interrupt flag</td>
</tr>
<tr>
<td></td>
<td>This bit is set when the START bit of DMA2D_CR, DMA2DFGFPCCCR or DMA2D_BGFPCCCR is set and a wrong configuration has been programmed.</td>
</tr>
<tr>
<td>29</td>
<td>CTCIF: CLUT transfer complete interrupt flag</td>
</tr>
<tr>
<td></td>
<td>This bit is set when the CLUT copy from a system memory area to the internal DMA2D memory is complete.</td>
</tr>
<tr>
<td>28</td>
<td>CAEIF: CLUT access error interrupt flag</td>
</tr>
<tr>
<td></td>
<td>This bit is set when the CPU accesses the CLUT while the CLUT is being automatically copied from a system memory to the internal DMA2D.</td>
</tr>
<tr>
<td>27</td>
<td>TWIF: Transfer watermark interrupt flag</td>
</tr>
<tr>
<td></td>
<td>This bit is set when the last pixel of the watermarked line has been transferred.</td>
</tr>
<tr>
<td>26</td>
<td>TCIF: Transfer complete interrupt flag</td>
</tr>
<tr>
<td></td>
<td>This bit is set when a DMA2D transfer operation is complete (data transfer only).</td>
</tr>
<tr>
<td>25</td>
<td>TEIF: Transfer error interrupt flag</td>
</tr>
<tr>
<td></td>
<td>This bit is set when an error occurs during a DMA transfer (data transfer or automatic CLUT loading).</td>
</tr>
</tbody>
</table>
18.6.3 DMA2D interrupt flag clear register (DMA2D_IFCR)

Address offset: 0x0008
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:6 Reserved, must be kept at reset value.

- **CCEIF**: Clear configuration error interrupt flag
  - Programming this bit to 1 clears the CEIF flag in the DMA2D_ISR register

- **CCTCIF**: Clear CLUT transfer complete interrupt flag
  - Programming this bit to 1 clears the CTCIF flag in the DMA2D_ISR register

- **CAECIF**: Clear CLUT access error interrupt flag
  - Programming this bit to 1 clears the CAEIF flag in the DMA2D_ISR register

- **CTWIF**: Clear transfer watermark interrupt flag
  - Programming this bit to 1 clears the TWIF flag in the DMA2D_ISR register

- **CTIF**: Clear transfer complete interrupt flag
  - Programming this bit to 1 clears the TCIF flag in the DMA2D_ISR register

- **CTEIF**: Clear transfer error interrupt flag
  - Programming this bit to 1 clears the TEIF flag in the DMA2D_ISR register
18.6.4 DMA2D foreground memory address register (DMA2D_FGMAR)

Address offset: 0x000C
Reset value: 0x0000 0000

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

Bits 31:0 **MA[31:0]: Memory address**
Address of the data used for the foreground image. This register can only be written when data transfers are disabled. Once the data transfer has started, this register is read-only.

The address alignment must match the image format selected e.g. a 32-bit per pixel format must be 32-bit aligned, a 16-bit per pixel format must be 16-bit aligned and a 4-bit per pixel format must be 8-bit aligned.

18.6.5 DMA2D foreground offset register (DMA2D_FGOR)

Address offset: 0x0010
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>LO[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **LO[15:0]: Line offset**
The line offset used for the foreground image, expressed in pixel when the LOM bit is reset and in byte when the LOM bit is set.
When expressed in pixels, only LO[13:0] bits are considered, LO[15:14] bits are ignored.
This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.
These bits can only be written when data transfers are disabled. Once the data transfer has started, they become read-only.
If the image format is 4-bit per pixel, the line offset must be even.
18.6.6 DMA2D background memory address register (DMA2D_BGMAR)

Address offset: 0x0014
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>MA[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 MA[31:0]: Memory address
Address of the data used for the background image. This register can only be written when data transfers are disabled. Once a data transfer has started, this register is read-only.
The address alignment must match the image format selected e.g. a 32-bit per pixel format must be 32-bit aligned, a 16-bit per pixel format must be 16-bit aligned and a 4-bit per pixel format must be 8-bit aligned.

18.6.7 DMA2D background offset register (DMA2D_BGOR)

Address offset: 0x0018
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 LO[15:0]: Line offset
The line offset used for the background image, expressed in pixel when the LOM bit is reset and in byte when the LOM bit is set.
When expressed in pixels, only LO[13:0] bits are considered, LO[15:14] bits are ignored.
This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.
These bits can only be written when data transfers are disabled. Once the data transfer has started, they become read-only.
If the image format is 4-bit per pixel, the line offset must be even.
18.6.8 DMA2D foreground PFC control register (DMA2D_FGPFCCR)

Address offset: 0x001C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHA[7:0]</td>
<td>Res.</td>
<td>Res.</td>
<td>RBS</td>
<td>AI</td>
<td>CSS[1:0]</td>
<td>AM[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS[7:0]</td>
<td>Res.</td>
<td>Res.</td>
<td>START</td>
<td>CCM</td>
<td>CM[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:24 ALPHA[7:0]: Alpha value**

These bits define a fixed alpha channel value which can replace the original alpha value or be multiplied by the original alpha value according to the alpha mode selected through the AM[1:0] bits.

These bits can only be written when data transfers are disabled. Once a transfer has started, they become read-only.

**Bits 23:22** Reserved, must be kept at reset value.

**Bit 21 RBS: Red Blue Swap**

This bit allows to swap the R & B to support BGR or ABGR color formats. Once the transfer has started, this bit is read-only.

0: Regular mode (RGB or ARGB)
1: Swap mode (BGR or ABGR)

**Bit 20 AI: Alpha Inverted**

This bit inverts the alpha value. Once the transfer has started, this bit is read-only.

0: Regular alpha
1: Inverted alpha

**Bits 19:18 CSS[1:0]: Chroma Sub-Sampling**

These bits define the chroma sub-sampling mode for YCbCr color mode. Once the transfer has started, these bits are read-only.

00: 4:4:4 (no chroma sub-sampling)
01: 4:2:2
10: 4:2:0
others: meaningless

**Bits 17:16 AM[1:0]: Alpha mode**

These bits select the alpha channel value to be used for the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, they become read-only.

00: No modification of the foreground image alpha channel value
01: Replace original foreground image alpha channel value by ALPHA[7: 0]
10: Replace original foreground image alpha channel value by ALPHA[7:0] multiplied with original alpha channel value
other configurations are meaningless
Bits 15:8 **CS[7:0]**: CLUT size

These bits define the size of the CLUT used for the foreground image. Once the CLUT transfer has started, this field is read-only.

The number of CLUT entries is equal to CS[7:0] + 1.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **START**: Start

This bit can be set to start the automatic loading of the CLUT. It is automatically reset:

- at the end of the transfer
- when the transfer is aborted by the user application by setting the ABORT bit in DMA2D_CR
- when a transfer error occurs
- when the transfer has not started due to a configuration error or another transfer operation already ongoing (data transfer or automatic background CLUT transfer).

Bit 4 **CCM**: CLUT color mode

This bit defines the color format of the CLUT. It can only be written when the transfer is disabled. Once the CLUT transfer has started, this bit is read-only.

0: ARGB8888
1: RGB888

Bits 3:0 **CM[3:0]**: Color mode

These bits define the color format of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

0000: ARGB8888
0001: RGB888
0010: RGB565
0011: ARGB1555
0100: ARGB4444
0101: L8
0110: AL44
0111: AL88
1000: L4
1001: A8
1010: A4
1011: YCbCr
others: meaningless
### 18.6.9 DMA2D foreground color register (DMA2D_FGCOLR)

Address offset: 0x0020  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>GREEN[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **RED[7:0]**: Red value

These bits defines the red value for the A4 or A8 mode of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Used also for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

Bits 15:8 **GREEN[7:0]**: Green value

These bits defines the green value for the A4 or A8 mode of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Used also for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

Bits 7:0 **BLUE[7:0]**: Blue value

These bits defines the blue value for the A4 or A8 mode of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Used also for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.
18.6.10  DMA2D background PFC control register (DMA2D_BGFCCR)

Address offset: 0x0024
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:24</th>
<th>ALPHA[7:0]: Alpha value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>These bits define a fixed alpha channel value which can replace the original alpha value or be multiplied with the original alpha value according to the alpha mode selected with bits AM[1:0]. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.</td>
</tr>
</tbody>
</table>

| Bits 23:22 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bit 21</th>
<th>RBS: Red Blue Swap</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Regular mode (RGB or ARGB)</td>
</tr>
<tr>
<td>1</td>
<td>Swap mode (BGR or ABGR)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 20</th>
<th>AI: Alpha Inverted</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Regular alpha</td>
</tr>
<tr>
<td>1</td>
<td>Inverted alpha</td>
</tr>
</tbody>
</table>

| Bits 19:18 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bits 17:16</th>
<th>AM[1:0]: Alpha mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>No modification of the foreground image alpha channel value</td>
</tr>
<tr>
<td>01</td>
<td>Replace original background image alpha channel value by ALPHA[7:0]</td>
</tr>
<tr>
<td>10</td>
<td>Replace original background image alpha channel value by ALPHA[7:0] multiplied with original alpha channel value</td>
</tr>
<tr>
<td>others</td>
<td>meaningless</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:8</th>
<th>CS[7:0]: CLUT size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CLUT size of 0 entries</td>
</tr>
<tr>
<td>others</td>
<td>CLUT size of 1 to 255 entries</td>
</tr>
</tbody>
</table>

| Bits 7:6   | Reserved, must be kept at reset value. |
Bit 5  **START**: Start
This bit is set to start the automatic loading of the CLUT. This bit is automatically reset:
- at the end of the transfer
- when the transfer is aborted by the user application by setting the ABORT bit in the DMA2D_CR
- when a transfer error occurs
- when the transfer has not started due to a configuration error or another transfer operation already on going (data transfer or automatic Background CLUT transfer).

Bit 4  **CCM**: CLUT Color mode
These bits define the color format of the CLUT. This register can only be written when the transfer is disabled. Once the CLUT transfer has started, this bit is read-only.
0: ARGB8888
1: RGB888

Bits 3:0  **CM[3:0]**: Color mode
These bits define the color format of the foreground image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.
0000: ARGB8888
0001: RGB888
0010: RGB565
0011: ARGB1555
0100: ARGB4444
0101: L8
0110: AL44
0111: AL88
1000: L4
1001: A8
1010: A4
others: meaningless
18.6.11 DMA2D background color register (DMA2D_BGCOLR)

Address offset: 0x0028
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23:16</td>
<td>RED[7:0]: Red value</td>
</tr>
<tr>
<td></td>
<td>These bits define the red value for the A4 or A8 mode of the background. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.</td>
</tr>
<tr>
<td></td>
<td>Used also for fixed color BG in memory-to-memory with blending and fixed color BG (FG fetch only with FG and BG PFC active) mode.</td>
</tr>
<tr>
<td>15:8</td>
<td>GREEN[7:0]: Green value</td>
</tr>
<tr>
<td></td>
<td>These bits define the green value for the A4 or A8 mode of the background. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.</td>
</tr>
<tr>
<td></td>
<td>Used also for fixed color BG in memory-to-memory with blending and fixed color BG (FG fetch only with FG and BG PFC active) mode.</td>
</tr>
<tr>
<td>7:0</td>
<td>BLUE[7:0]: Blue value</td>
</tr>
<tr>
<td></td>
<td>These bits define the blue value for the A4 or A8 mode of the background. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.</td>
</tr>
<tr>
<td></td>
<td>Used also for fixed color BG in memory-to-memory with blending and fixed color BG (FG fetch only with FG and BG PFC active) mode.</td>
</tr>
</tbody>
</table>

18.6.12 DMA2D foreground CLUT memory address register (DMA2D_FGCMAR)

Address offset: 0x002C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>MA[31:16]</td>
</tr>
<tr>
<td></td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15:0</td>
<td>MA[15:0]</td>
</tr>
<tr>
<td></td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>
18.6.13 DMA2D background CLUT memory address register (DMA2D_BGCMAR)

Address offset: 0x0030
Reset value: 0x0000 0000

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

Bits 31:0 MA[31:0]: Memory Address
Address of the data used for the CLUT address dedicated to the foreground image. This register can only be written when no transfer is ongoing. Once the CLUT transfer has started, this register is read-only.
If the foreground CLUT format is 32-bit, the address must be 32-bit aligned.

18.6.14 DMA2D output PFC control register (DMA2D_OPFCCR)

Address offset: 0x0034
Reset value: 0x0000 0000

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

Bits 31:0 MA[31:0]: Memory address
Address of the data used for the CLUT address dedicated to the background image. This register can only be written when no transfer is ongoing. Once the CLUT transfer has started, this register is read-only.
If the background CLUT format is 32-bit, the address must be 32-bit aligned.
Bits 31:22  Reserved, must be kept at reset value.

Bit 21 **RBS**: Red Blue Swap
This bit allows to swap the R & B to support BGR or ABGR color formats. Once the transfer has started, this bit is read-only.
0: Regular mode (RGB or ARGB)
1: Swap mode (BGR or ABGR)

Bit 20 **AI**: Alpha Inverted
This bit inverts the alpha value. Once the transfer has started, this bit is read-only.
0: Regular alpha
1: Inverted alpha

Bits 19:9  Reserved, must be kept at reset value.

Bit 8 **SB**: Swap Bytes
When set, the bytes in the output FIFO are swapped two by two.
When this bit is set, the number of pixel per line (PL) must be even, and the output memory address (OMAR) must be even.
This register can only be written when the transfer is disabled. Once the transfer has started, this register is read-only.
0: Bytes in regular order in the output FIFO
1: Bytes are swapped two by two in the output FIFO

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

Bits 2:0 **CM[2:0]**: Color mode
These bits define the color format of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.
000: ARGB8888
001: RGB888
010: RGB565
011: ARGB1555
100: ARGB4444
others: meaningless

### 18.6.15 DMA2D output color register (DMA2D_OCOLR)

Address offset: 0x0038
Reset value: 0x0000 0000

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

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

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

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

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

### 704/2974  RM0455 Rev 3
ARGB8888 or RGB888 color mode

Bits 31:24  **ALPHA[7:0]**: Alpha channel value in ARGB8888 mode otherwise reserved
These bits define the alpha channel of the output color. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 23:16  **RED[7:0]**: Red value in ARGB8888 or RGB888 mode otherwise reserved
These bits define the red value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 15:8  **GREEN[7:0]**: Green value in ARGB8888 or RGB888
These bits define the green value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 7:0  **BLUE[7:0]**: Blue value in ARGB8888 or RGB888
These bits define the blue value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

RGB565 color mode

Bits 15:11  **RED[4:0]**: Red value in RGB565 mode
These bits define the red value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 12:5  **GREEN[5:0]**: Green value in RGB565 mode
These bits define the green value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 4:0  **BLUE[4:0]**: Blue value in RGB565 mode
These bits define the blue value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

ARGB1555 color mode

Bit 15  **A**: Alpha channel value in ARGB1555 mode
This bit defines the alpha channel of the output color. This bit can only be written when data transfers are disabled. Once the transfer has started, it is read-only.

Bits 14:10  **RED[4:0]**: Red value in ARGB1555 mode
These bits define the red value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 9:5  **GREEN[4:0]**: Green value in ARGB1555 mode
These bits define the green value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 4:0  **BLUE[4:0]**: Blue value in ARGB1555 mode
These bits define the blue value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

ARGB4444 color mode

Bits 15:12  **ALPHA[3:0]**: Alpha channel value in ARGB4444
This bit defines the alpha channel of the output color. This bit can only be written when data transfers are disabled. Once the transfer has started, it is read-only.
Bits 11:8 **RED[3:0]**: Red value in ARGB4444 mode
These bits define the red value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 7:4 **GREEN[3:0]**: Green value in ARGB4444 mode
These bits define the green value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 3:0 **BLUE[3:0]**: Blue value in ARGB4444 mode
These bits define the blue value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

### 18.6.16 DMA2D output memory address register (DMA2D_OMAR)

**Address offset:** 0x003C  
**Reset value:** 0x0000 0000

![](image)

Bits 31:0 **MA[31:0]**: Memory address  
Address of the data used for the output FIFO. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.  
The address alignment must match the image format selected e.g. a 32-bit per pixel format must be 32-bit aligned and a 16-bit per pixel format must be 16-bit aligned.

### 18.6.17 DMA2D output offset register (DMA2D_OOR)

**Address offset:** 0x0040  
**Reset value:** 0x0000 0000

![](image)
Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  LO[15:0]: Line offset
          The line offset used for the output expressed in pixel when the LOM bit is reset and in
          byte when the LOM bit is set.
          When expressed in pixels, only LO[13:0] is considered, LO[15:14] are ignored.
          This value is used for the address generation. It is added at the end of each line to
          determine the starting address of the next line.
          These bits can only be written when data transfers are disabled. Once data transfer has
          started, they become read-only.

18.6.18  DMA2D number of line register (DMA2D_NLR)

Address offset: 0x0044
Reset value: 0x0000 0000

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

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

Bits 29:16  PL[13:0]: Pixel per lines
          Number of pixels per lines of the area to be transferred. These bits can only be written
          when data transfers are disabled. Once the transfer has started, they are read-only.
          If any of the input image format is 4-bit per pixel, pixel per lines must be even.

Bits 15:0  NL[15:0]: Number of lines
          Number of lines of the area to be transferred. These bits can only be written when data
          transfers are disabled. Once the transfer has started, they are read-only.

18.6.19  DMA2D line watermark register (DMA2D_LWR)

Address offset: 0x0048
Reset value: 0x0000 0000

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

LW[15:0]

rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw
18.6.20 DMA2D AXI master timer configuration register (DMA2D_AMTCR)

Address offset: 0x004C
Reset value: 0x0000 0000

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

- Bits 31:16 Reserved, must be kept at reset value.
- Bits 15:0 **LW[15:0]**: Line watermark
  These bits allow to configure the line watermark for interrupt generation.
  An interrupt is raised when the last pixel of the watermarked line has been transferred.
  These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

- Bits 7:0 **DT[7:0]**: Dead time
  Dead time value in the AXI clock cycle inserted between two consecutive accesses on
  the AXI master port. These bits represent the minimum guaranteed number of cycles
  between two consecutive AXI accesses.

- Bit 0 **EN**: Enable
  Enables the dead time functionality.

18.6.21 DMA2D foreground CLUT (DMA2D_FGCLUT[y])

Address offset: 0x0400 + 4*y, (y = 0 to 255)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHA&lt;y&gt;[7:0]</td>
<td>RED&lt;y&gt;[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREEN&lt;y&gt;[7:0]</td>
<td>BLUE&lt;y&gt;[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
18.6.22 DMA2D background CLUT (DMA2D_BGCLUT[y])

Address offset: 0x0800 + 4*y, (y = 0 to 255)
Reset value: 0xXXXX XXXX

| Bits 31:24 | ALPHA<y>[7:0]: Alpha <y> |
| Alpha value for index <y> for the foreground. |
| Bits 23:16 | RED<y>[7:0]: Red <y> |
| Red value for index <y> for the foreground. |
| Bits 15:8 | GREEN<y>[7:0]: Green <y> |
| Green value for index <y> for the foreground. |
| Bits 7:0 | BLUE<y>[7:0]: Blue <y> |
| Blue value for index <y> for the foreground. |
18.6.23 DMA2D register map

The following table summarizes the DMA2D registers. Refer to Section 2.3 on page 128 for the DMA2D register base address.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>MA[31:0]</th>
<th>Mode[2:0]</th>
<th>CEB</th>
<th>CTGE</th>
<th>CAE</th>
<th>TWE</th>
<th>TCE</th>
<th>TEE</th>
<th>LO</th>
<th>ABORT</th>
<th>SUSP</th>
<th>START</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>DMA2D_CR</td>
<td>000</td>
<td>000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0004</td>
<td>DMA2D_ISR</td>
<td>000000</td>
<td>000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0008</td>
<td>DMA2D_IFCR</td>
<td>000000</td>
<td>000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x000C</td>
<td>DMA2D_FGMAR</td>
<td>00000000</td>
<td>00000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0010</td>
<td>DMA2D_FGOR</td>
<td>00000000</td>
<td>00000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0014</td>
<td>DMA2D_BGMAR</td>
<td>00000000</td>
<td>00000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0018</td>
<td>DMA2D_BGOR</td>
<td>00000000</td>
<td>00000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x001C</td>
<td>DMA2D_FGPFCCR</td>
<td>00000000</td>
<td>00000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0020</td>
<td>DMA2D_FGCOLR</td>
<td>00000000</td>
<td>00000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0024</td>
<td>DMA2D_BGPFCCR</td>
<td>00000000</td>
<td>00000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0028</td>
<td>DMA2D_BGCOLR</td>
<td>00000000</td>
<td>00000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x002C</td>
<td>DMA2D_FGCMAR</td>
<td>00000000</td>
<td>00000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0030</td>
<td>DMA2D_BGCMAR</td>
<td>00000000</td>
<td>00000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0034</td>
<td>DMA2D_OPFCCR</td>
<td>00000000</td>
<td>00000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 122. DMA2D register map and reset values
Refer to Section 2.3 on page 128 for the register boundary addresses.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x003B</td>
<td>DMA2D_OCOLR</td>
<td>0x0040</td>
<td>DMA2D_OOR</td>
</tr>
<tr>
<td>0x003C</td>
<td>DMA2D_OMAR</td>
<td>0x0044</td>
<td>DMA2D_NLR</td>
</tr>
<tr>
<td>0x0040</td>
<td>DMA2D_OOR</td>
<td>0x004B</td>
<td>DMA2D_LWR</td>
</tr>
<tr>
<td>0x0044</td>
<td>DMA2D_NLR</td>
<td>0x004C</td>
<td>DMA2D_AMTCR</td>
</tr>
<tr>
<td>0x0050-0x003FC</td>
<td>Reserved</td>
<td>0x004E</td>
<td>DMA2D_FGCLUT</td>
</tr>
<tr>
<td>0x0050-0x007FC</td>
<td>Reserved</td>
<td>0x004F</td>
<td>DMA2D_BGCLUT</td>
</tr>
<tr>
<td>0x0050-0x00FFC</td>
<td>Reserved</td>
<td>0x0050-0x00FFC</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 122. DMA2D register map and reset values (continued)
19 Nested vectored interrupt controller (NVIC)

19.1 NVIC features

The NVIC includes the following features:

- up to 154 maskable interrupt channels for STM32H7A3/7B3 and STM32H7B0 (not including the 16 interrupt lines of Cortex®-M7 with FPU)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- low-latency exception and interrupt handling
- power management control
- implementation of system control registers

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts, including the core exceptions, are managed by the NVIC.

For more information on exceptions and NVIC programming, refer to PM0253 programming manual for Cortex®-M7.

19.1.1 SysTick calibration value register

The SysTick calibration value (SYST_CALIB) is fixed to 0x3E8. It provides a reference timebase of 1 ms based when the SysTick clock frequency is 1 MHz. To match the 1 ms timebase whatever the application frequency, the SysTick reload value must be programmed as follows in the SYST_RVR register:

- The SysTick clock source is the 100 MHz CPU clock (HCLK):

  \[ \text{reload value} = (F_{\text{HCLK}} \times \text{SYST_CALIB}) - 1 \]

- or the SysTick clock source is an external clock:

  \[ \text{reload value} = \left(\frac{F_{\text{HCLK}}}{8} \times \text{SYST_CALIB}\right) - 1 \]

  where \( F_{\text{HCLK}} \) refers to the AHB frequency expressed in MHz.

For example, to achieve a timebase of 1 ms when the SysTick clock source is the 100 MHz HCLK:

\[ \text{reload value} = (100 \times \text{SYST_CALIB}) - 1 = 0x1869F \]

19.1.2 Interrupt and exception vectors

The exception vectors connected to the NVIC are the following: reset, NMI, HardFault, MemManage, Bus Fault, UsageFault, SVCall, DebugMonitor, PendSV, SysTick.
Table 123. NVIC(1)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0000</td>
</tr>
<tr>
<td>-</td>
<td>-3</td>
<td>-</td>
<td>Reset</td>
<td>0x0000 0004</td>
</tr>
<tr>
<td>-</td>
<td>-2</td>
<td>NMI</td>
<td>Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.</td>
<td>0x0000 0008</td>
</tr>
<tr>
<td>-</td>
<td>-1</td>
<td>HardFault</td>
<td>All classes of fault</td>
<td>0x0000 000C</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>MemManage</td>
<td>Memory management</td>
<td>0x0000 0010</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>BusFault</td>
<td>Prefetch fault, memory access fault</td>
<td>0x0000 0014</td>
</tr>
<tr>
<td>-</td>
<td>2</td>
<td>UsageFault</td>
<td>Undefined instruction or illegal state</td>
<td>0x0000 0018</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 001C-</td>
</tr>
<tr>
<td>-</td>
<td>3</td>
<td>SVCalls</td>
<td>System service call via SWI instruction</td>
<td>0x0000 002B</td>
</tr>
<tr>
<td>-</td>
<td>4</td>
<td>DebugMonitor</td>
<td>Debug monitor</td>
<td>0x0000 0030</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0034</td>
</tr>
<tr>
<td>-</td>
<td>5</td>
<td>PendSV</td>
<td>Pendable request for system service</td>
<td>0x0000 0038</td>
</tr>
<tr>
<td>-</td>
<td>6</td>
<td>SysTick</td>
<td>System tick timer</td>
<td>0x0000 003C</td>
</tr>
<tr>
<td>wwdg_it</td>
<td>7</td>
<td>WWGD</td>
<td>Window Watchdog interrupt</td>
<td>0x0000 0040</td>
</tr>
<tr>
<td>exti_pwr_pvd_wkup</td>
<td>8</td>
<td>PVD_PVM</td>
<td>PVD through EXTI line detection interrupt</td>
<td>0x0000 0044</td>
</tr>
<tr>
<td>exti_tamp_RTC_wkup</td>
<td>9</td>
<td>RTC_TAMP_STAMP_CSS_LSE</td>
<td>RTC tamper, timestamp via CSS LSE</td>
<td>0x0000 0048</td>
</tr>
<tr>
<td>lsecss_rcc_it</td>
<td>10</td>
<td>RTC_WKUP</td>
<td>RTC Wakeup interrupt through the EXTI line</td>
<td>0x0000 004C</td>
</tr>
<tr>
<td>flash_it</td>
<td>11</td>
<td>FLASH</td>
<td>Flash memory global interrupt</td>
<td>0x0000 0050</td>
</tr>
<tr>
<td>rcc_it</td>
<td>12</td>
<td>RCC</td>
<td>RCC global interrupt</td>
<td>0x0000 0054</td>
</tr>
<tr>
<td>exti_exti0_wkup</td>
<td>13</td>
<td>EXTI0</td>
<td>EXTI Line 0 interrupt</td>
<td>0x0000 0058</td>
</tr>
<tr>
<td>exti_exti1_wkup</td>
<td>14</td>
<td>EXTI1</td>
<td>EXTI Line 1 interrupt</td>
<td>0x0000 005C</td>
</tr>
<tr>
<td>exti_exti2_wkup</td>
<td>15</td>
<td>EXTI2</td>
<td>EXTI Line 2 interrupt</td>
<td>0x0000 0060</td>
</tr>
<tr>
<td>exti_exti3_wkup</td>
<td>16</td>
<td>EXTI3</td>
<td>EXTI Line 3 interrupt</td>
<td>0x0000 0064</td>
</tr>
<tr>
<td>exti_exti4_wkup</td>
<td>17</td>
<td>EXTI4</td>
<td>EXTI Line 4 interrupt</td>
<td>0x0000 0068</td>
</tr>
<tr>
<td>dma1_it0</td>
<td>18</td>
<td>DMA1_STR0</td>
<td>DMA1 Stream0 global interrupt</td>
<td>0x0000 006C</td>
</tr>
</tbody>
</table>
### Table 123. NVIC\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Priority</th>
<th>NVIC position</th>
<th>Acronym</th>
<th>Description</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>dma1_it1</td>
<td>19</td>
<td>12</td>
<td>DMA1_STR1</td>
<td>DMA1 Stream1 global interrupt</td>
<td>0x0000 0070</td>
</tr>
<tr>
<td>dma1_it2</td>
<td>20</td>
<td>13</td>
<td>DMA1_STR2</td>
<td>DMA1 Stream2 global interrupt</td>
<td>0x0000 0074</td>
</tr>
<tr>
<td>dma1_it3</td>
<td>21</td>
<td>14</td>
<td>DMA1_STR3</td>
<td>DMA1 Stream3 global interrupt</td>
<td>0x0000 0078</td>
</tr>
<tr>
<td>dma1_it4</td>
<td>22</td>
<td>15</td>
<td>DMA1_STR4</td>
<td>DMA1 Stream4 global interrupt</td>
<td>0x0000 007C</td>
</tr>
<tr>
<td>dma1_it5</td>
<td>23</td>
<td>16</td>
<td>DMA1_STR5</td>
<td>DMA1 Stream5 global interrupt</td>
<td>0x0000 0080</td>
</tr>
<tr>
<td>dma1_it6</td>
<td>24</td>
<td>17</td>
<td>DMA1_STR6</td>
<td>DMA1 Stream6 global interrupt</td>
<td>0x0000 0084</td>
</tr>
<tr>
<td>adc1_it</td>
<td>25</td>
<td>18</td>
<td>ADC1_2</td>
<td>ADC1 and ADC2 global interrupt</td>
<td>0x0000 0088</td>
</tr>
<tr>
<td>adc2_it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ttfdcan_intr0_it</td>
<td>26</td>
<td>19</td>
<td>FDCAN1_IT0</td>
<td>TTCAN Interrupt 0</td>
<td>0x0000 008C</td>
</tr>
<tr>
<td>fdcan_intr0_it</td>
<td>27</td>
<td>20</td>
<td>FDCAN2_IT0</td>
<td>FDCAN Interrupt 0</td>
<td>0x0000 0090</td>
</tr>
<tr>
<td>ttfdcan_intr1_it</td>
<td>28</td>
<td>21</td>
<td>FDCAN1_IT1</td>
<td>TTCAN Interrupt 1</td>
<td>0x0000 0094</td>
</tr>
<tr>
<td>fdcan_intr1_it</td>
<td>29</td>
<td>22</td>
<td>FDCAN2_IT1</td>
<td>FDCAN Interrupt 1</td>
<td>0x0000 0098</td>
</tr>
<tr>
<td>exti_exti5_wkup</td>
<td>30</td>
<td>23</td>
<td>EXTI9_5</td>
<td>EXTI Line[9:5] interrupts</td>
<td>0x0000 009C</td>
</tr>
<tr>
<td>exti_exti6_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exti_exti7_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exti_exti8_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exti_exti9_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim1_brk_it</td>
<td>31</td>
<td>24</td>
<td>TIM1_BRK</td>
<td>TIM1 break interrupt</td>
<td>0x0000 00A0</td>
</tr>
<tr>
<td>tim1_upd_it</td>
<td>32</td>
<td>25</td>
<td>TIM1_UP</td>
<td>TIM1 update interrupt</td>
<td>0x0000 00A4</td>
</tr>
<tr>
<td>tim1_trg_it</td>
<td>33</td>
<td>26</td>
<td>TIM1_TRG_COM</td>
<td>TIM1 trigger and commutation</td>
<td>0x0000 00A8</td>
</tr>
<tr>
<td>tim1_cc_it</td>
<td>34</td>
<td>27</td>
<td>TIM1_CC</td>
<td>TIM1 capture / compare interrupt</td>
<td>0x0000 00AC</td>
</tr>
<tr>
<td>tim2_it</td>
<td>35</td>
<td>28</td>
<td>TIM2</td>
<td>TIM2 global interrupt</td>
<td>0x0000 00B0</td>
</tr>
<tr>
<td>tim3_it</td>
<td>36</td>
<td>29</td>
<td>TIM3</td>
<td>TIM3 global interrupt</td>
<td>0x0000 00B4</td>
</tr>
<tr>
<td>tim4_it</td>
<td>37</td>
<td>30</td>
<td>TIM4</td>
<td>TIM4 global interrupt</td>
<td>0x0000 00B8</td>
</tr>
<tr>
<td>i2c1_ev_it</td>
<td>38</td>
<td>31</td>
<td>I2C1_EV</td>
<td>I2C1 event interrupt</td>
<td>0x0000 00BC</td>
</tr>
<tr>
<td>exti_i2c1_ev_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i2c1_err_it</td>
<td>39</td>
<td>32</td>
<td>I2C1_ER</td>
<td>I2C1 error interrupt</td>
<td>0x0000 00C0</td>
</tr>
</tbody>
</table>
Table 123. NVIC\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Priority</th>
<th>NVIC position</th>
<th>Acronym</th>
<th>Description</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>i2c2_ev_it</td>
<td>40</td>
<td>33</td>
<td>I2C2_EV</td>
<td>I2C2 event interrupt</td>
<td>0x0000 00C4</td>
</tr>
<tr>
<td>exti_i2c2_ev_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i2c2_err_it</td>
<td>41</td>
<td>34</td>
<td>I2C2_ER</td>
<td>I2C2 error interrupt</td>
<td>0x0000 00C8</td>
</tr>
<tr>
<td>spi1_it</td>
<td>42</td>
<td>35</td>
<td>SPI1</td>
<td>SPI1 global interrupt</td>
<td>0x0000 00CC</td>
</tr>
<tr>
<td>exti_spi1_it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spi2_it</td>
<td>43</td>
<td>36</td>
<td>SPI2</td>
<td>SPI2 global interrupt</td>
<td>0x0000 00D0</td>
</tr>
<tr>
<td>exti_spi2_it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>usart1_gbl_it</td>
<td>44</td>
<td>37</td>
<td>USART1</td>
<td>USART1 global interrupt</td>
<td>0x0000 00D4</td>
</tr>
<tr>
<td>exti_usart1_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>usart2_gbl_it</td>
<td>45</td>
<td>38</td>
<td>USART2</td>
<td>USART2 global interrupt</td>
<td>0x0000 00D8</td>
</tr>
<tr>
<td>exti_usart2_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>usart3_gbl_it</td>
<td>46</td>
<td>39</td>
<td>USART3</td>
<td>USART3 global interrupt</td>
<td>0x0000 00DC</td>
</tr>
<tr>
<td>exti_usart3_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exti_exti10_wkup</td>
<td>47</td>
<td>40</td>
<td>EXTI15_10</td>
<td>EXTI Line[15:10] interrupts</td>
<td>0x0000 00E0</td>
</tr>
<tr>
<td>exti_exti11_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exti_exti12_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exti_exti13_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exti_exti14_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exti_exti15_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exti_rtc_al</td>
<td>48</td>
<td>41</td>
<td>RTC_ALARM</td>
<td>RTC alarms (A and B) through EXTI Line interrupts</td>
<td>0x0000 00E4</td>
</tr>
<tr>
<td>dfsdm2_it</td>
<td>49</td>
<td>42</td>
<td>DFSDM2</td>
<td>DFSDM2 interrupt</td>
<td>0x0000 00E8</td>
</tr>
<tr>
<td>tim8_brk_it</td>
<td>50</td>
<td>43</td>
<td>TIM8_BRK_TIM12</td>
<td>TIM8 break and TIM12 global interrupts</td>
<td>0x0000 00EC</td>
</tr>
<tr>
<td>tim12_gbl_it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim8_upd_it</td>
<td>51</td>
<td>44</td>
<td>TIM8_UP_TIM13</td>
<td>TIM8 update and TIM13 global interrupts</td>
<td>0x0000 00F0</td>
</tr>
<tr>
<td>tim13_gbl_it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim8_trg_it</td>
<td>52</td>
<td>45</td>
<td>TIM8_TRG_COM_TIM14</td>
<td>TIM8 trigger /commutation and TIM14 global interrupts</td>
<td>0x0000 00F4</td>
</tr>
<tr>
<td>tim14_gbl_it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim8_cc_it</td>
<td>53</td>
<td>46</td>
<td>TIM8_CC</td>
<td>TIM8 capture / compare interrupts</td>
<td>0x0000 00F8</td>
</tr>
<tr>
<td>dma1_it7</td>
<td>54</td>
<td>47</td>
<td>DMA1_STR7</td>
<td>DMA1 Stream7 global interrupt</td>
<td>0x0000 00FC</td>
</tr>
<tr>
<td>fmc_gbl_it</td>
<td>55</td>
<td>48</td>
<td>FMC</td>
<td>FMC global interrupt</td>
<td>0x0000 0100</td>
</tr>
<tr>
<td>sdmmc_it</td>
<td>56</td>
<td>49</td>
<td>SDMMC1</td>
<td>SDMMC global interrupt</td>
<td>0x0000 0104</td>
</tr>
</tbody>
</table>
### Table 123. NVIC\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Priority</th>
<th>NVIC position</th>
<th>Acronym</th>
<th>Description</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim5_gbl_it</td>
<td>57</td>
<td>50</td>
<td>TIM5</td>
<td>TIM5 global interrupt</td>
<td>0x0000 0108</td>
</tr>
<tr>
<td>spi3_it</td>
<td>58</td>
<td>51</td>
<td>SPI3</td>
<td>SPI3 global interrupt</td>
<td>0x0000 010C</td>
</tr>
<tr>
<td>exti_spi3_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uart4_gbl_it</td>
<td>59</td>
<td>52</td>
<td>UART4</td>
<td>UART4 global interrupt</td>
<td>0x0000 0110</td>
</tr>
<tr>
<td>exti_uart4_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uart5_gbl_it</td>
<td>60</td>
<td>53</td>
<td>UART5</td>
<td>UART5 global interrupt</td>
<td>0x0000 0114</td>
</tr>
<tr>
<td>exti_uart5_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim6_gbl_it</td>
<td>61</td>
<td>54</td>
<td>TIM6_DAC1</td>
<td>TIM6 global interrupt</td>
<td>0x0000 0118</td>
</tr>
<tr>
<td>dac1_unr_it</td>
<td></td>
<td></td>
<td></td>
<td>DAC1 underrun error interrupt</td>
<td>0x0000 0134</td>
</tr>
<tr>
<td>tim7_gbl_it</td>
<td>62</td>
<td>55</td>
<td>TIM7</td>
<td>TIM7 global interrupt</td>
<td>0x0000 011C</td>
</tr>
<tr>
<td>dma2_it0</td>
<td>63</td>
<td>56</td>
<td>DMA2_STR0</td>
<td>DMA2 Stream0 interrupt</td>
<td>0x0000 0120</td>
</tr>
<tr>
<td>dma2_it1</td>
<td>64</td>
<td>57</td>
<td>DMA2_STR1</td>
<td>DMA2 Stream1 interrupt</td>
<td>0x0000 0124</td>
</tr>
<tr>
<td>dma2_it2</td>
<td>65</td>
<td>58</td>
<td>DMA2_STR2</td>
<td>DMA2 Stream2 interrupt</td>
<td>0x0000 0128</td>
</tr>
<tr>
<td>dma2_it3</td>
<td>66</td>
<td>59</td>
<td>DMA2_STR3</td>
<td>DMA2 Stream3 interrupt</td>
<td>0x0000 012C</td>
</tr>
<tr>
<td>dma2_it4</td>
<td>67</td>
<td>60</td>
<td>DMA2_STR4</td>
<td>DMA2 Stream4 interrupt</td>
<td>0x0000 0130</td>
</tr>
<tr>
<td></td>
<td>68</td>
<td></td>
<td></td>
<td>Reserved</td>
<td>0x0000 0134</td>
</tr>
<tr>
<td></td>
<td>69</td>
<td></td>
<td></td>
<td>Reserved</td>
<td>0x0000 0138</td>
</tr>
<tr>
<td>fdcan_cal_it</td>
<td>70</td>
<td>63</td>
<td>FDCAN_CAL</td>
<td>FDCAN calibration interrupts</td>
<td>0x0000 013C</td>
</tr>
<tr>
<td>dfsdm1_it4</td>
<td>71</td>
<td>64</td>
<td>DFSDM1_FLT4</td>
<td>DFSDM1 filter 4 interrupt</td>
<td>0x0000 0140</td>
</tr>
<tr>
<td>dfsdm1_it5</td>
<td>72</td>
<td>65</td>
<td>DFSDM1_FLT5</td>
<td>DFSDM1 filter 5 interrupt</td>
<td>0x0000 0144</td>
</tr>
<tr>
<td>dfsdm1_it6</td>
<td>73</td>
<td>66</td>
<td>DFSDM1_FLT6</td>
<td>DFSDM1 filter 6 interrupt</td>
<td>0x0000 0148</td>
</tr>
<tr>
<td>dfsdm1_it7</td>
<td>74</td>
<td>67</td>
<td>DFSDM1_FLT7</td>
<td>DFSDM1 filter 7 interrupt</td>
<td>0x0000 014C</td>
</tr>
<tr>
<td>dma2_it5</td>
<td>75</td>
<td>68</td>
<td>DMA2_STR5</td>
<td>DMA2 Stream5 interrupt</td>
<td>0x0000 0150</td>
</tr>
<tr>
<td>dma2_it6</td>
<td>76</td>
<td>69</td>
<td>DMA2_STR6</td>
<td>DMA2 Stream6 interrupt</td>
<td>0x0000 0154</td>
</tr>
<tr>
<td>dma2_it7</td>
<td>77</td>
<td>70</td>
<td>DMA2_STR7</td>
<td>DMA2 Stream7 interrupt</td>
<td>0x0000 0158</td>
</tr>
<tr>
<td>usart6_gbl_it</td>
<td>78</td>
<td>71</td>
<td>USART6</td>
<td>USART6 global interrupt</td>
<td>0x0000 015C</td>
</tr>
<tr>
<td>exti_usart6_wkup</td>
<td></td>
<td></td>
<td></td>
<td>USART6 wakeup interrupt</td>
<td></td>
</tr>
<tr>
<td>i2c3_ev_it</td>
<td>79</td>
<td>72</td>
<td>I2C3_EV</td>
<td>2C3 event interrupt</td>
<td>0x0000 0160</td>
</tr>
<tr>
<td>exti_i2c3_ev_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i2c3_err_it</td>
<td>80</td>
<td>73</td>
<td>I2C3_ER</td>
<td>2C3 error interrupt</td>
<td>0x0000 0164</td>
</tr>
<tr>
<td>usb1_out_it</td>
<td>81</td>
<td>74</td>
<td>OTG_HS_EP1_OUT</td>
<td>OTG_HS out global interrupt</td>
<td>0x0000 0168</td>
</tr>
<tr>
<td>usb1_in_it</td>
<td>82</td>
<td>75</td>
<td>OTG_HS_EP1_IN</td>
<td>OTG_HS in global interrupt</td>
<td>0x0000 016C</td>
</tr>
<tr>
<td>exti_usb1_wkup</td>
<td>83</td>
<td>76</td>
<td>OTG_HS_WKUP</td>
<td>OTG_HS wakeup interrupt</td>
<td>0x0000 0170</td>
</tr>
</tbody>
</table>
Table 123. NVIC\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Priority</th>
<th>NVIC position</th>
<th>Acronym</th>
<th>Description</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>usb1_gbl_it</td>
<td>84</td>
<td>77</td>
<td>OTG_HS</td>
<td>OTG_HS global interrupt</td>
<td>0x0000 0174</td>
</tr>
<tr>
<td>dcmi_it/pssi_it</td>
<td>85</td>
<td>78</td>
<td>DCMI/PSSI</td>
<td>DCMI/PSSI global interrupt</td>
<td>0x0000 0178</td>
</tr>
<tr>
<td>cryp_it</td>
<td>86</td>
<td>79</td>
<td>CRYP</td>
<td>CRYP global interrupt</td>
<td>0x0000 017C</td>
</tr>
<tr>
<td>hash_rng_it</td>
<td>87</td>
<td>80</td>
<td>HASH_RNG</td>
<td>HASH and RNG global interrupt</td>
<td>0x0000 0180</td>
</tr>
<tr>
<td>cpu_fpu_it</td>
<td>88</td>
<td>81</td>
<td>FPU</td>
<td>CPU FPU interrupt</td>
<td>0x0000 0184</td>
</tr>
<tr>
<td>uart7_gbl_it</td>
<td>89</td>
<td>82</td>
<td>UART7</td>
<td>UART7 global interrupt</td>
<td>0x0000 0188</td>
</tr>
<tr>
<td>exti_uart7_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uart8_gbl_it</td>
<td>90</td>
<td>83</td>
<td>UART8</td>
<td>UART8 global interrupt</td>
<td>0x0000 018C</td>
</tr>
<tr>
<td>exti_uart8_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spi4_it</td>
<td>91</td>
<td>84</td>
<td>SPI4</td>
<td>SPI4 global interrupt</td>
<td>0x0000 0190</td>
</tr>
<tr>
<td>exti_spi4_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spi5_it</td>
<td>92</td>
<td>85</td>
<td>SPI5</td>
<td>SPI5 global interrupt</td>
<td>0x0000 0194</td>
</tr>
<tr>
<td>exti_spi5_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spi6_it</td>
<td>93</td>
<td>86</td>
<td>SPI6</td>
<td>SPI6 global interrupt</td>
<td>0x0000 0198</td>
</tr>
<tr>
<td>exti_spi6_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sai1_it</td>
<td>94</td>
<td>87</td>
<td>SAI1</td>
<td>SAI1 global interrupt</td>
<td>0x0000 019C</td>
</tr>
<tr>
<td>ltdc_it</td>
<td>95</td>
<td>88</td>
<td>LTDC</td>
<td>LCD-TFT global interrupt</td>
<td>0x0000 01A0</td>
</tr>
<tr>
<td>ltdc_err_it</td>
<td>96</td>
<td>89</td>
<td>LTDC_ER</td>
<td>LCD-TFT error interrupt</td>
<td>0x0000 01A4</td>
</tr>
<tr>
<td>dma2d_gbl_it</td>
<td>97</td>
<td>90</td>
<td>DMA2D</td>
<td>DMA2D global interrupt</td>
<td>0x0000 01A8</td>
</tr>
<tr>
<td>sai2_it</td>
<td>98</td>
<td>91</td>
<td>SAI2</td>
<td>SAI2 global interrupt</td>
<td>0x0000 01AC</td>
</tr>
<tr>
<td>octospi1_it</td>
<td>99</td>
<td>92</td>
<td>OCTOSPI1</td>
<td>OCTOSPI global interrupt</td>
<td>0x0000 01B0</td>
</tr>
<tr>
<td>lptim1_it</td>
<td>100</td>
<td>93</td>
<td>LPTIM1</td>
<td>LPTIM1 global interrupt</td>
<td>0x0000 01B4</td>
</tr>
<tr>
<td>exti_lptim_wkup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cec_it</td>
<td>101</td>
<td>94</td>
<td>CEC</td>
<td>HDMI-CEC global interrupt</td>
<td>0x0000 01B8</td>
</tr>
<tr>
<td>exti_cec_it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i2c4_ev_it</td>
<td>102</td>
<td>95</td>
<td>I2C4_EV</td>
<td>2C4 event interrupt</td>
<td>0x0000 01BC</td>
</tr>
<tr>
<td>exti_i2c4_ev_it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i2c4_err_it</td>
<td>103</td>
<td>96</td>
<td>I2C4_ER</td>
<td>2C4 error interrupt</td>
<td>0x0000 01C0</td>
</tr>
<tr>
<td>spdifrx_it</td>
<td>104</td>
<td>97</td>
<td>SPDIFRX</td>
<td>SPDIFRX global interrupt</td>
<td>0x0000 01C4</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td>0x0000 01C8</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td>0x0000 01CC</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td>0x0000 01D0</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td>0x0000 01D4</td>
</tr>
</tbody>
</table>
Table 123. NVIC\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Priority</th>
<th>NVIC position</th>
<th>Acronym</th>
<th>Description</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>dmamux1_ovr_it</td>
<td>109</td>
<td>102</td>
<td>DMAMUX1_OV</td>
<td>DMAMUX1 overrun interrupt</td>
<td>0x0000 01D8</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 01DC</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 01E0</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 01E4</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 01E8</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 01EC</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 01F0</td>
</tr>
<tr>
<td>dfsdm1_it0</td>
<td>117</td>
<td>110</td>
<td>DFSDM1_FLT0</td>
<td>DFSDM1 filter 0 interrupt</td>
<td>0x0000 01F8</td>
</tr>
<tr>
<td>dfsdm1_it1</td>
<td>118</td>
<td>111</td>
<td>DFSDM1_FLT1</td>
<td>DFSDM1 filter 1 interrupt</td>
<td>0x0000 01FC</td>
</tr>
<tr>
<td>dfsdm1_it2</td>
<td>119</td>
<td>112</td>
<td>DFSDM1_FLT2</td>
<td>DFSDM1 filter 2 interrupt</td>
<td>0x0000 0200</td>
</tr>
<tr>
<td>dfsdm1_it3</td>
<td>120</td>
<td>113</td>
<td>DFSDM1_FLT3</td>
<td>DFSDM1 filter 3 interrupt</td>
<td>0x0000 0204</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0208</td>
</tr>
<tr>
<td>swpmi_gbl_it</td>
<td>122</td>
<td>115</td>
<td>SWPMI1</td>
<td>SWPMI global interrupt</td>
<td>0x0000 020C</td>
</tr>
<tr>
<td>exti_swpmi_wup</td>
<td>122</td>
<td>115</td>
<td>SWPMI1</td>
<td>SWPMI wakeup</td>
<td>0x0000 020C</td>
</tr>
<tr>
<td>tim15_gbl_it</td>
<td>123</td>
<td>116</td>
<td>TIM15</td>
<td>TIM15 global interrupt</td>
<td>0x0000 0210</td>
</tr>
<tr>
<td>tim16_gbl_it</td>
<td>124</td>
<td>117</td>
<td>TIM16</td>
<td>TIM16 global interrupt</td>
<td>0x0000 0214</td>
</tr>
<tr>
<td>tim17_gbl_it</td>
<td>125</td>
<td>118</td>
<td>TIM17</td>
<td>TIM17 global interrupt</td>
<td>0x0000 0218</td>
</tr>
<tr>
<td>exti_mdios_wkup</td>
<td>126</td>
<td>119</td>
<td>MDTOS_WKUP</td>
<td>MDIOS wakeup</td>
<td>0x0000 021C</td>
</tr>
<tr>
<td>mdios_it</td>
<td>127</td>
<td>120</td>
<td>MDTOS</td>
<td>MDTOS global interrupt</td>
<td>0x0000 0220</td>
</tr>
<tr>
<td>jpeg_it</td>
<td>128</td>
<td>121</td>
<td>JPEG</td>
<td>JPEG global interrupt</td>
<td>0x0000 0224</td>
</tr>
<tr>
<td>mdma_it</td>
<td>129</td>
<td>122</td>
<td>MDMA</td>
<td>MDMA</td>
<td>0x0000 0228</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 022C</td>
</tr>
<tr>
<td>sdmmc2_it</td>
<td>131</td>
<td>124</td>
<td>SDMMC2</td>
<td>SDMMC2 global interrupt</td>
<td>0x0000 0230</td>
</tr>
<tr>
<td>hsem1_it</td>
<td>132</td>
<td>125</td>
<td>HSEM0</td>
<td>HSEM global interrupt 1</td>
<td>0x0000 0234</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0238</td>
</tr>
<tr>
<td>dac2_unr_it</td>
<td>134</td>
<td>127</td>
<td>DAC2</td>
<td>DAC2 underrun interrupt</td>
<td>0x0000 023C</td>
</tr>
<tr>
<td>dmamux2_ovr_it</td>
<td>135</td>
<td>128</td>
<td>DMAMUX2_OVR</td>
<td>DMAMUX2 overrun interrupt</td>
<td>0x0000 0240</td>
</tr>
<tr>
<td>bdma2_ch0_it</td>
<td>136</td>
<td>129</td>
<td>BDMA2_CH0</td>
<td>BDMA2 channel 0 interrupt</td>
<td>0x0000 0244</td>
</tr>
<tr>
<td>bdma2_ch1_it</td>
<td>137</td>
<td>130</td>
<td>BDMA2_CH1</td>
<td>BDMA2 channel 1 interrupt</td>
<td>0x0000 0248</td>
</tr>
<tr>
<td>bdma2_ch2_it</td>
<td>138</td>
<td>131</td>
<td>BDMA2_CH2</td>
<td>BDMA2 channel 2 interrupt</td>
<td>0x0000 024C</td>
</tr>
<tr>
<td>bdma2_ch3_it</td>
<td>139</td>
<td>132</td>
<td>BDMA2_CH3</td>
<td>BDMA2 channel 3 interrupt</td>
<td>0x0000 0250</td>
</tr>
<tr>
<td>bdma2_ch4_it</td>
<td>140</td>
<td>133</td>
<td>BDMA2_CH4</td>
<td>BDMA2 channel 4 interrupt</td>
<td>0x0000 0254</td>
</tr>
<tr>
<td>Signal</td>
<td>Priority</td>
<td>NVIC position</td>
<td>Acronym</td>
<td>Description</td>
<td>Address offset</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
<td>---------------</td>
<td>---------</td>
<td>------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>bdma2_ch5_it</td>
<td>141</td>
<td>134</td>
<td>BDMA2_CH5</td>
<td>BDMA2 channel 5 interrupt</td>
<td>0x0000 0258</td>
</tr>
<tr>
<td>bdma_ch6_it</td>
<td>142</td>
<td>135</td>
<td>BDMA_CH6</td>
<td>BDMA channel 6 interrupt</td>
<td>0x0000 025C</td>
</tr>
<tr>
<td>bdma_ch7_it</td>
<td>143</td>
<td>136</td>
<td>BDMA_CH7</td>
<td>BDMA channel 7 interrupt</td>
<td>0x0000 0260</td>
</tr>
<tr>
<td>comp_gbl_it</td>
<td>144</td>
<td>137</td>
<td>COMP</td>
<td>COMP1 and COMP2 global interrupt</td>
<td>0x0000 0264</td>
</tr>
<tr>
<td>exti_comp1_wkup</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 027C</td>
</tr>
<tr>
<td>exti_comp2_wkup</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0280</td>
</tr>
<tr>
<td>lptim2_it</td>
<td>145</td>
<td>138</td>
<td>LPTIM2</td>
<td>LPTIM2 timer interrupt</td>
<td>0x0000 0268</td>
</tr>
<tr>
<td>exti_lptim2_wkup</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0284</td>
</tr>
<tr>
<td>lptim3_it</td>
<td>146</td>
<td>139</td>
<td>LPTIM3</td>
<td>LPTIM2 timer interrupt</td>
<td>0x0000 026C</td>
</tr>
<tr>
<td>exti_lptim3_wkup</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0288</td>
</tr>
<tr>
<td>uart9_it</td>
<td>147</td>
<td>140</td>
<td>UART9</td>
<td>UART9 global interrupt</td>
<td>0x0000 0270</td>
</tr>
<tr>
<td>exti_uart9_wkup</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0290</td>
</tr>
<tr>
<td>usart10_it</td>
<td>148</td>
<td>141</td>
<td>USART10</td>
<td>USART10 global interrupt</td>
<td>0x0000 0274</td>
</tr>
<tr>
<td>exti_usart10_wkup</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0294</td>
</tr>
<tr>
<td>lpuart_gbl_it</td>
<td>149</td>
<td>142</td>
<td>LPUART1</td>
<td>LPUART1 global interrupt</td>
<td>0x0000 0278</td>
</tr>
<tr>
<td>exti_lpuart_rx_it</td>
<td></td>
<td></td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 028C</td>
</tr>
<tr>
<td>exti_lpuart_tx_it</td>
<td></td>
<td></td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0290</td>
</tr>
<tr>
<td>crs_it</td>
<td>151</td>
<td>144</td>
<td>CRS</td>
<td>Clock Recovery System global interrupt</td>
<td>0x0000 0280</td>
</tr>
<tr>
<td>ecc_it</td>
<td>152</td>
<td>145</td>
<td>ECC</td>
<td>ECC diagnostic global interrupt</td>
<td>0x0000 0284</td>
</tr>
<tr>
<td>exti_dts_wkup</td>
<td>154</td>
<td>147</td>
<td>DTS_IT</td>
<td>Temperature sensor global interrupt</td>
<td>0x0000 0288</td>
</tr>
<tr>
<td>dts_it</td>
<td>155</td>
<td>148</td>
<td>DTS_IT</td>
<td>Temperature sensor global interrupt</td>
<td>0x0000 0290</td>
</tr>
<tr>
<td>exti_wkup1_wkup</td>
<td>156</td>
<td>149</td>
<td>WKUP</td>
<td>WKUP0 to WKUP5 pins</td>
<td>0x0000 0294</td>
</tr>
<tr>
<td>octospi2_it</td>
<td>157</td>
<td>150</td>
<td>OCTOSPI2</td>
<td>OCTOSPI2 global interrupt</td>
<td>0x0000 0298</td>
</tr>
<tr>
<td>otfdec1_sec_it</td>
<td>158</td>
<td>151</td>
<td>OTFDEC1</td>
<td>OTFDEC1 interrupt</td>
<td>0x0000 029C</td>
</tr>
<tr>
<td>otfdec2_sec_it</td>
<td>159</td>
<td>152</td>
<td>OTFDEC2</td>
<td>OTFDEC2 interrupt</td>
<td>0x0000 02A0</td>
</tr>
</tbody>
</table>
1. When different signals are connected to the same NVIC interrupt line, they are OR-ed.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Priority</th>
<th>NVIC position</th>
<th>Acronym</th>
<th>Description</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>gfxmmu_it</td>
<td>160</td>
<td>153</td>
<td>GFXMMU</td>
<td>GFXMMU interrupt</td>
<td>0x0000 02A4</td>
</tr>
<tr>
<td>bdma1_it</td>
<td>161</td>
<td>154</td>
<td>BDMA1</td>
<td>BDMA1 global interrupt</td>
<td>0x0000 02A8</td>
</tr>
</tbody>
</table>

Table 123. NVIC\(^{(1)}\) (continued)
20 Extended interrupt and event controller (EXTI)

The Extended Interrupt and event controller (EXTI) manages wakeup through configurable and direct event inputs. It provides wakeup requests to the power controller (PWR), and generates interrupt requests to the CPU NVIC and to the SRD domain DMAMUX2, and events to the CPU event input.

The EXTI wakeup requests allow the system to be woken up from Stop mode, and the CPU to be woken up from CStop mode.

In addition, both the interrupt request and event request generation can be used in Run mode.

20.1 EXTI main features

The EXTI main features are the following:

- All event inputs allow the CPU to wake up and to generate a CPU interrupt and/or a CPU event
- Some event inputs allow the user to wake up the SRD domain for Autonomous run mode and generate an interrupt to the SRD domain, i.e. the DMAMUX2

The asynchronous event inputs are classified in two groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse), that have the following features:
  - Selectable active trigger edge
  - Interrupt pending status register bit
  - Individual Interrupt and event generation mask
  - Software trigger capability
  - Configurable system SRD domain wakeup events.
    They have an SRD pending mask and a status register, and may have an SRD interrupt signal.

- Direct events (interrupt and wakeup sources from other peripherals, requiring to be cleared in the peripheral)

  Direct events feature:
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI (the interrupt pending status is provided by the peripheral generating the event)
  - Individual interrupt and event generation mask
  - No software trigger capability
  - Direct system SRD domain wakeup events have an SRD pending mask and status register, and may have an SRD interrupt signal
20.2 EXTI block diagram

As shown in Figure 93, the EXTI consists of a register block accessed via an APB interface, an event input trigger block, and a masking block.

The register block contains all EXTI registers.

The event input trigger block provides event input edge triggering logic.

The masking block provides the event input distribution to the different wakeup, interrupt and event outputs, as well as their masking.

Figure 93. EXTI block diagram

20.2.1 EXTI connections between peripherals, CPU and SRD domain

The peripherals able to generate wakeup events when the system is in Stop mode or the CPU is in CStop mode are connected to an EXTI configurable event input or direct event input:

- Peripheral signals that generate a pulse are connected to an EXTI configurable event input. For these events, the EXTI provides a CPU status pending bit that has to be cleared.

- Peripheral interrupt and wakeup sources that have to be cleared in the peripheral are connected to an EXTI direct event input. There is no CPU status pending bit within the EXTI. The Interrupt or Wakeup is cleared by the CPU in the peripheral.

The event inputs able to wakeup SRD for Autonomous run mode are provided with an SRD domain pending request function, that has to be cleared. This clearing request is taken care of by the signal selected by the pending clear selection.

The CPU interrupts are connected to their respective CPU NVIC. Similarly, the CPU event is connected to the CPU rxev input.
The EXTI wakeup signals are connected to the PWR block. They are used to wake up the SRD domain and/or the CPU.

The SRD domain interrupts allow the system to trigger events for SRD domain Autonomous run operations.

20.3 EXTI functional description

Different logic implementations can be used, depending on the EXTI event input type and wakeup target(s). The applicable features are controlled from register bits:

- Active trigger edge enable, by EXTI rising trigger selection register (EXTI_RTSR1), EXTI rising trigger selection register (EXTI_RTSR2), EXTI rising trigger selection register (EXTI_RTSR3), and EXTI falling trigger selection register (EXTI_FTSR1), EXTI falling trigger selection register (EXTI_FTSR2), EXTI falling trigger selection register (EXTI_FTSR3)

- Software trigger, by EXTI software interrupt event register (EXTI_SWIER1), EXTI software interrupt event register (EXTI_SWIER2), EXTI software interrupt event register (EXTI_SWIER3)

- CPU Interrupt enable, by EXTI interrupt mask register (EXTI_CPUIMR1), EXTI interrupt mask register (EXTI_CPUIMR2), EXTI interrupt mask register (EXTI_CPUIMR3)

- CPU Event enable, by EXTI event mask register (EXTI_CPUEMR1), EXTI event mask register (EXTI_CPUEMR2), EXTI event mask register (EXTI_CPUEMR3)

- SRD domain wakeup pending, by EXTI SRD pending mask register (EXTI_SRDPMR1), EXTI SRD pending mask register (EXTI_SRDPMR2), EXTI SRD pending mask register (EXTI_SRDPMR3)

<table>
<thead>
<tr>
<th>Event Input Type</th>
<th>Wakeup Target(s)</th>
<th>Logic Implementation</th>
<th>EXTI_RTSR</th>
<th>EXTI_FTSR</th>
<th>EXTI_SWIER</th>
<th>EXTI_CPUIMR</th>
<th>EXTI_CPUEMR</th>
<th>EXTI_SRDPMR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configurable</td>
<td>CPU</td>
<td>Configurable event input, CPU wakeup logic</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Any(2)</td>
<td>Configurable event input, Any wakeup logic</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Direct</td>
<td>CPU</td>
<td>Direct event input, CPU wakeup logic</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Any(2)</td>
<td>Direct event input, Any wakeup logic</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>X</td>
</tr>
</tbody>
</table>

1. X indicates that the functionality is available.

2. Waking up SRD domain for Autonomous run mode and/or CPU.
20.3.1 EXTI configurable event input CPU wakeup

*Figure 95* provides a detailed representation of the logic associated to the configurable event inputs that always wake up the CPU.

*Figure 94. Configurable event triggering logic CPU wakeup*

The software interrupt event register allows the system to trigger configurable events by software, writing the EXTI software interrupt event register (EXTI_SWIER1), the EXTI software interrupt event register (EXTI_SWIER2), or the EXTI software interrupt event register (EXTI_SWIER3) register bit.

The rising edge EXTI rising trigger selection register (EXTI_RTSR1), EXTI rising trigger selection register (EXTI_RTSR2), EXTI rising trigger selection register (EXTI_RTSR3), and falling edge EXTI falling trigger selection register (EXTI_FTSR1), EXTI falling trigger selection register (EXTI_FTSR2), EXTI falling trigger selection register (EXTI_FTSR3) selection registers allow the system to enable and select the configurable event active trigger edge or both edges.

The devices feature dedicated interrupt mask registers, namely EXTI interrupt mask register (EXTI_CPUIMR1) and EXTI interrupt mask register (EXTI_CPUIMR2), EXTI interrupt mask register (EXTI_CPUIMR3), and EXTI pending register (EXTI_CPUPR1), EXTI pending register (EXTI_CPUPR2), EXTI pending register (EXTI_CPUPR3) for configurable events pending request registers. The CPU pending register is only set for an unmasked CPU interrupt. Each event provides a individual CPU interrupt to the CPU NVIC. The configurable event interrupts need to be acknowledged by software in the EXTI_CPUPR register.
The devices feature dedicated event mask registers, i.e. EXTI event mask register (EXTI_CPUEMR1), EXTI event mask register (EXTI_CPUEMR2), and EXTI event mask register (EXTI_CPUEMR3). The enabled event then generates an event on the CPU. All events for a CPU are OR-ed together into a single CPU event signal. The CPU pending register (EXTI_CPUPR) is not set for an unmasked CPU event.

When a CPU interrupt or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the CPU clock is woken up before the asynchronous edge detection circuit is reset.

**Note:** A detected configurable event, enabled by the CPU, is only cleared when the CPU wakes up.

### 20.3.2 EXTI configurable event input Any wakeup

*Figure 95* provides a detailed description of the logic associated to the configurable event inputs that can wake up SRD domain for Autonomous run mode and/or CPU (“Any” target). It provides the same functionality as the configurable event input CPU wakeup, with additional functionality to wake up the SRD domain independently.

When all CPU interrupts and CPU events are disabled, the asynchronous edge detection circuit is reset by the SRD domain clocked delay and rising edge detect pulse generator. This guarantees that the SRD domain clock is woken up before the asynchronous edge detection circuit is reset.

<table>
<thead>
<tr>
<th>EXTI_C1IMR</th>
<th>EXTI_C1EMR</th>
<th>Asynchronous Edge detector reset by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both = 0</td>
<td></td>
<td>SRD domain clock rising edge detect pulse generator</td>
</tr>
<tr>
<td>At least one = 1</td>
<td></td>
<td>CPU clock rising edge detect pulse generator</td>
</tr>
</tbody>
</table>
The event triggering logic for “Any” target has additional SRD pending mask register EXTI SRD pending mask register (EXTI_SRDPMR1), EXTI SRD pending mask register (EXTI_SRDPMR2), EXTI SRD pending mask register (EXTI_SRDPMR3) and SRD pending request logic. The SRD pending request logic is only set for unmasked SRD pending events. The SRD pending request logic keeps the SRD domain in Run mode until the SRD pending request logic is cleared by the selected SRD domain pendclear source.
20.3.3 EXTI direct event input CPU wakeup

*Figure 96* provides a detailed description of the logic associated with the direct event inputs that wake up the CPU.

Direct events only provide CPU interrupt enable and CPU event enable functionality.

1. The CPU interrupt for asynchronous direct event inputs (peripheral wakeup signals) is synchronized with the CPU clock. The synchronous direct event inputs (peripheral interrupt signals), after the asynchronous edge detection, are directly sent to the CPU interrupt without resynchronization.
20.3.4 EXTI direct event input Any wakeup

Figure 97 provides a detailed description of the logic associated to the direct event inputs that wake up SRD domain for Autonomous run mode and/or CPU, (“Any” target). It provides the same functionality as the direct event input CPU wakeup, plus additional functionality to wakeup the SRD domain independently.

Figure 97. Direct event triggering logic Any wakeup

---

1. The CPU interrupt and SRD domain interrupt for asynchronous direct event inputs (peripheral wakeup signals) are synchronized with the CPU clock and the SRD domain clock, respectively. The synchronous direct event inputs (peripheral interrupt signals), after the asynchronous edge detection, are directly sent to the CPU interrupt and the SRD domain interrupt without resynchronization in the EXTI.
20.3.5 EXTI SRD pending request clear selection

The SRD pending request logic of the event inputs that are able to wake up SRD domain for Autonomous run mode can be cleared by the selected SRD pendclear source.

For each SRD pending request, an SRD domain pendclear source can be selected from four different inputs. 

*Figure 98* provides a detailed description of the SRD pendclear source selection logic.

![Figure 98. SRD domain pending request clear logic](image)

The SRD pending request clear selection registers *EXTI SRD pending clear selection register low (EXTI_SRDPCR1L)*, *EXTI SRD pending clear selection register high (EXTI_SRDPCR1H)*, *EXTI SRD pending clear selection register low (EXTI_SRDPCR2L)*, *EXTI SRD pending clear selection register high (EXTI_SRDPCR2H)* and *EXTI SRD pending clear selection register high (EXTI_SRDPCR3H)* allow the system to select the source to reset the SRD pending request.

20.4 EXTI event input mapping

For the sixteen GPIO event inputs, the associated GPIO pin has to be selected in the SYSCFG_EXTICRn register. The same pin from each GPIO maps to the corresponding EXTI event input.

The wakeup capabilities of each event input are detailed in *Table 126*. An event input can wake up the CPU, and in the case of “Any” can also wake up SRD domain for Autonomous run mode.

The EXTI event inputs that are connected to the CPU NVIC are indicated in the *Connection to NVIC* column. For the EXTI events that do not have a connection to the NVIC, the peripheral interrupt is directly connected to the NVIC in parallel with the connection to the EXTI.

All EXTI event inputs are OR-ed together and connected to the CPU event input (rxev).
<table>
<thead>
<tr>
<th>Event input</th>
<th>Source</th>
<th>Event input type</th>
<th>Wakeup target(s)</th>
<th>Connection to NVIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 15</td>
<td>EXTI[15:0]</td>
<td>Configurable</td>
<td>Any</td>
<td>Yes</td>
</tr>
<tr>
<td>16</td>
<td>PVD and AVD&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>Configurable</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>17</td>
<td>RTC alarms</td>
<td>Configurable</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>18</td>
<td>RTC tamper, RTC timestamp,</td>
<td>Configurable</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>RCC LSECSS&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>RTC wakeup timer</td>
<td>Configurable</td>
<td>Any</td>
<td>Yes</td>
</tr>
<tr>
<td>20</td>
<td>COMP1</td>
<td>Configurable</td>
<td>Any</td>
<td>Yes</td>
</tr>
<tr>
<td>21</td>
<td>COMP2</td>
<td>Configurable</td>
<td>Any</td>
<td>Yes</td>
</tr>
<tr>
<td>22</td>
<td>I2C1 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>23</td>
<td>I2C2 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>24</td>
<td>I2C3 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>25</td>
<td>I2C4 wakeup</td>
<td>Direct</td>
<td>Any</td>
<td>Yes</td>
</tr>
<tr>
<td>26</td>
<td>USART1 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>27</td>
<td>USART2 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>28</td>
<td>USART3 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>29</td>
<td>USART6 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>30</td>
<td>UART4 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>31</td>
<td>UART5 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>32</td>
<td>UART7 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>33</td>
<td>UART8 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>34</td>
<td>LPUART1 RX wakeup</td>
<td>Direct</td>
<td>Any</td>
<td>Yes</td>
</tr>
<tr>
<td>35</td>
<td>LPUART1 TX wakeup</td>
<td>Direct</td>
<td>Any</td>
<td>Yes</td>
</tr>
<tr>
<td>36</td>
<td>SPI1 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>37</td>
<td>SPI2 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>38</td>
<td>SPI3 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>39</td>
<td>SPI4 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>40</td>
<td>SPI5 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>41</td>
<td>SPI6 wakeup</td>
<td>Direct</td>
<td>Any</td>
<td>Yes</td>
</tr>
<tr>
<td>42</td>
<td>MDIO wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>43</td>
<td>USB1 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>44</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>45</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>46</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>47</td>
<td>LPTIM1 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>48</td>
<td>LPTIM2 wakeup</td>
<td>Direct</td>
<td>Any</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Table 126. EXTI Event input mapping (continued)

<table>
<thead>
<tr>
<th>Event input</th>
<th>Source</th>
<th>Event input type</th>
<th>Wakeup target(s)</th>
<th>Connection to NVIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>49</td>
<td>LPTIM2 output</td>
<td>Configurable</td>
<td>Any</td>
<td>No(3)</td>
</tr>
<tr>
<td>50</td>
<td>LPTIM3 wakeup</td>
<td>Direct</td>
<td>Any</td>
<td>Yes</td>
</tr>
<tr>
<td>51</td>
<td>LPTIM3 output</td>
<td>Configurable</td>
<td>Any</td>
<td>No(3)</td>
</tr>
<tr>
<td>52</td>
<td>UART9 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>53</td>
<td>USART10 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>54</td>
<td>SWPMI1 wakeup</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>55(4)</td>
<td>WKUP1</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>56(4)</td>
<td>WKUP2</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>57(4)</td>
<td>WKUP3</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>58(4)</td>
<td>WKUP4</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>59(4)</td>
<td>WKUP5</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>60(4)</td>
<td>WKUP6</td>
<td>Direct</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>61</td>
<td>RCC interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>62</td>
<td>I2C4 Event interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>63</td>
<td>I2C4 Error interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>64</td>
<td>LPUART1 global Interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>65</td>
<td>SPI6 interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>66</td>
<td>BDMA2 CH0 interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>67</td>
<td>BDMA2 CH1 interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>68</td>
<td>BDMA2 CH2 interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>69</td>
<td>BDMA2 CH3 interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>70</td>
<td>BDMA2 CH4 interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>71</td>
<td>BDMA2 CH5 interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>72</td>
<td>BDMA2 CH6 interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>73</td>
<td>BDMA2 CH7 interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>74</td>
<td>DMAMUX2 interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>75</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>76</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>77</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>78</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>79</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>80</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>81</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>82</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>83</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
20.5 EXTI functional behavior

The direct event inputs are enabled in the respective peripheral that generating the event. The configurable events are enabled by enabling at least one of the trigger edges.

In Stop mode, an event always wakes up the SRD domain.

In system Run and Stop modes, an event always generates an associated SRD domain interrupt. An event only wakes up the CPU when the event associated CPU interrupt is unmasked and/or the CPU event is unmasked.

For configurable event inputs, an event request is generated when the enabled edge(s) is/are detected on the event input. When the associated CPU interrupt is unmasked, the corresponding pending PRx bit in EXTI_CPUPR is set and the CPU interrupt signal is activated. EXTI_CPUPR PRx pending bit must be cleared by software by programming it to 1. This clears the CPU interrupt.

For direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit. When the associated CPU interrupt is unmasked, the corresponding CPU interrupt signal is activated.

---

Table 126. EXTI Event input mapping (continued)

<table>
<thead>
<tr>
<th>Event input</th>
<th>Source</th>
<th>Event input type</th>
<th>Wakeup target(s)</th>
<th>Connection to NVIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>84</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>85</td>
<td>HDMI-CEC wakeup</td>
<td>Configurable</td>
<td>CPU only</td>
<td>Yes</td>
</tr>
<tr>
<td>86</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>87</td>
<td>HSECSS interrupt</td>
<td>Direct</td>
<td>CPU only</td>
<td>No(5)</td>
</tr>
<tr>
<td>88</td>
<td>TEMP wakeup</td>
<td>Direct</td>
<td>Any</td>
<td>Yes</td>
</tr>
</tbody>
</table>

1. PVD and AVD signals are OR-ed together on the same EXTI event input.
2. RTC Tamper, RTC timestamp and RCC LSESSL signals are OR-ed together on the same EXTI event input.
3. Not available on CPU NVIC, to be used for system wakeup only or CPU event input (rxev).
4. Signals of WKUP1 to WKUP6 correspond to WKUPn pin+1.
5. Available on CPU NVIC directly from the peripheral.

Table 127. Masking functionality

<table>
<thead>
<tr>
<th>CPU</th>
<th>Event enable MRx bits of EXTI_CPUEMR</th>
<th>Configurable event inputs PRx bits of EXTI_CPUPR</th>
<th>SRD domain wakeup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Interrupt enable MRx bits of EXTI_CPUIMR</td>
<td>Interrupt</td>
<td>Event</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>No</td>
<td>Masked</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>No</td>
<td>Masked</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Status latched</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Status latched</td>
<td>Yes</td>
</tr>
</tbody>
</table>

1. Only for event inputs that allow the system to wake up SRD domain for Autonomous run mode (Any target).
2. For event inputs that always wake up the CPU.

---

For configurable event inputs, an event request is generated when the enabled edge(s) is/are detected on the event input. When the associated CPU interrupt is unmasked, the corresponding pending PRx bit in EXTI_CPUPR is set and the CPU interrupt signal is activated. EXTI_CPUPR PRx pending bit must be cleared by software by programming it to 1. This clears the CPU interrupt.

For direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit. When the associated CPU interrupt is unmasked, the corresponding CPU interrupt signal is activated.
The CPU event has to be unmasked to generate an event. When the enabled edge(s) is/are detected on the event input, a CPU event pulse is generated. There is no CPU event pending bit.

Both a CPU interrupt and a CPU event may be enabled on the same event input. They both trigger the same event input condition(s).

For configurable event inputs, an event input request can be generated by software by writing 1 in the software interrupt/event register EXTI_SWIER.

Whenever an event input is enabled and a CPU interrupt and/or CPU event is unmasked, the event input also generates an SRD domain wakeup next to the CPU wakeup.

Some event inputs can wake up the SRD domain Autonomous run mode. In this case the CPU interrupt and CPU event are masked, preventing the CPU to be woken up. Two SRD domain Autonomous run mode wakeup mechanisms are supported:

- **SRD domain wakeup without pending (EXTI_SRDPMR = 0)**
  - On a configurable event input, this mechanism wakes up SRD domain and automatically clears the SRD domain wakeup signal after the delay + rising edge detect pulse generator.
  - On a direct event input, this mechanism wakes up SRD domain and clears the SRD domain wakeup signal after the direct event input signal is cleared.

- **SRD domain wakeup with pending (EXTI_SRDPMR = 1)**
  - On a configurable event input, this mechanism wakes up SRD domain and clears the SRD domain wakeup signal after the delay + rising edge detect pulse generator and when the SRD pending request is cleared.
  - On a direct event input, this mechanism wakes up SRD domain and clears the SRD domain wakeup signal after the direct event input signal is cleared and when the SRD pending request is cleared.

### 20.5.1 EXTI CPU interrupt procedure

1. Unmask the event input interrupt by setting the corresponding mask bits in the EXTI_CPUIMR register.
2. For configurable event inputs, enable the event input by setting either one or both the corresponding trigger edge enable bits in EXTI_RTSR and EXTI_FTSR registers.
3. Enable the associated interrupt source in the CPU NVIC or use the SEVONPEND, so that an interrupt coming from the CPU interrupt signal can be detected by the CPU after a WFI/WFE instruction.
   
   For configurable event inputs, the associated EXTI pending bit needs to be cleared.

### 20.5.2 EXTI CPU event procedure

1. Unmask the event input by setting the corresponding mask bits of the EXTI_CPUEMR register.
2. For configurable event inputs, enable the event input by setting either one or both the corresponding trigger edge enable bits in EXTI_RTSR and EXTI_FTSR registers.
3. The CPU event signal is detected by the CPU after a WFE instruction.
   
   For configurable event inputs, there is no EXTI pending bit to clear.
20.5.3 EXTI CPU wakeup procedure
1. Unmask the event input by setting at least one of the corresponding mask bits in the EXTI_CPUIMR and/or EXTI_CPUEMR registers. The CPU wakeup is generated at the same time as the unmasked CPU interrupt and/or CPU event.
2. For configurable event inputs, enable the event input by setting either one or both the corresponding trigger edge enable bits in EXTI_RTSR and EXTI_FTSR registers. Direct events automatically generate a CPU wakeup.

20.5.4 EXTI SRD domain wakeup for Autonomous run mode procedure
1. Mask the event input for waking up the CPU by clearing both the corresponding mask bits in the EXTI_CPUIMR and/or EXTI_CPUEMR registers.
2. For configurable event inputs, enable the event input by setting either one or both the corresponding trigger edge enable bits in EXTI_RTSR and EXTI_FTSR registers.
3. Direct events automatically generate an SRD domain wakeup.
4. Select the SRD domain wakeup mechanism in EXTI_SRDPMR.
   – When SRD domain wakeup without pending (EXTI_PMR = 0) is selected, the Wakeup is automatically cleared following the clearing of the event input.
   – When SRD domain wakeup with pending (EXTI_PMR = 1) is selected, the Wakeup needs to be cleared by a selected SRD domain pendclear source.
     A pending SRD domain wakeup signal can also be cleared by firmware by clearing the associated EXTI_SRDPMR register bit.
5. An SRD domain interrupt is generated after the SRD domain wakeup:
   – Configurable event inputs generate a pulse on SRD domain interrupt.
   – Direct event inputs activate the SRD domain interrupt until the event input is cleared in the peripheral.

20.5.5 EXTI software interrupt/event trigger procedure
Any of the configurable event inputs can be triggered from the software interrupt/event register (the associated CPU interrupt and/or CPU event shall be enabled by their respective procedure). Follow the steps below:
1. Enable the event input by setting at least one of the corresponding edge trigger bits in the EXTI_RTSR and/or EXTI_FTSR registers.
2. Unmask the software interrupt/event trigger by setting at least one of the corresponding mask bits in the EXTI_CPUIMR and/or EXTI_CPUEMR registers.
3. Trigger the software interrupt/event by writing 1 to the corresponding bit in the EXTI_SWIER register.
4. The event input can be disabled by clearing the EXTI_RTSR and EXTI_FTSR register bits.

Note: An edge on the configurable event input also triggers an interrupt/event.
A software trigger can be used to set the SRD pending request logic, keeping the SRD domain in Run until the SRD pending request logic is cleared.
20.6 EXTI registers

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

20.6.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x00

Reset value: 0x0000 0000

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

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

Bits 21:0 TR[21:0]: Rising trigger event configuration bit of configurable event input x (x= 21 to 0)

0: Rising trigger disabled (for event and Interrupt) for input line
1: Rising trigger enabled (for event and Interrupt) for input line

Note: The configurable event inputs are edge triggered, no glitch must be generated on these inputs.

If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

20.6.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

Bits 21:0 TR[21:0]: Falling trigger event configuration bit of configurable event input x (x= 21 to 0)

0: Rising trigger disabled (for event and Interrupt) for input line
1: Rising trigger enabled (for event and Interrupt) for input line

Note: The configurable event inputs are edge triggered, no glitch must be generated on these inputs.

If a falling edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.
Bits 31:22  Reserved, must be kept at reset value.

Bits 21:0  **TR[21:0]:** Falling trigger event configuration bit of configurable event input x (x= 21 to 0)
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

**Note:** The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same Configurable event input. In this case, both edges generate a trigger.

### 20.6.3 EXTI software interrupt event register (EXTI_SWIERT1)

Address offset: 0x08
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>SWIER15</th>
<th>SWIER14</th>
<th>SWIER13</th>
<th>SWIER12</th>
<th>SWIER11</th>
<th>SWIER10</th>
<th>SWIER 9</th>
<th>SWIER 8</th>
<th>SWIER 7</th>
<th>SWIER 6</th>
<th>SWIER 5</th>
<th>SWIER 4</th>
<th>SWIER 3</th>
<th>SWIER 2</th>
<th>SWIER 1</th>
<th>SWIER 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bits 21:0  **SWIER[21:0]:** Software interrupt on line x (x = 21 to 0)
This bitfield always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit will trigger an event on line x.
This bit is automatically cleared by hardware.

### 20.6.4 EXTI SRD pending mask register (EXTI_SRDPMR1)

Address offset: 0x0C
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:26  Reserved, must be kept at reset value.

Bit 25  **MR25**: SRD pending mask on event input x (x = 25)
0: SRD pending request from line x is masked. Writing this bit to 0 also clears the SRD Pending request.
1: SRD pending request from line x is unmasked. When triggered, the SRD domain pending signal keeps SRD domain wakeup active until cleared.

Bits 24:22  Reserved, must be kept at reset value.

Bits 21:19  **MR[21:19]**: SRD pending mask on event input x (x = 21 to 19)
0: SRD pending request from line x is masked. Writing this bit to 0 also clears the SRD pending request.
1: SRD pending request from line x is unmasked. When triggered, the SRD domain pending signal keeps SRD domain wakeup active until cleared.

Bits 18:16  Reserved, must be kept at reset value.

Bits 15:0  **MR[15:0]**: SRD pending mask on event input x (x = 15 to 0)
0: SRD pending request from line x is masked. Writing this bit to 0 also clears the SRD pending request.
1: SRD pending request from line x is unmasked. When triggered, the SRD domain pending signal keeps SRD domain wakeup active until cleared.

### 20.6.5  EXTI SRD pending clear selection register low (EXTI_SRDP PCR1L)

Address offset: 0x10

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>PCS15</th>
<th>PCS14</th>
<th>PCS13</th>
<th>PCS12</th>
<th>PCS11</th>
<th>PCS10</th>
<th>PCS9</th>
<th>PCS8</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15 14</td>
<td>13 12</td>
<td>11 10</td>
<td>9 8</td>
<td>7 6</td>
<td>5 4</td>
<td>3 2</td>
<td>1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PCS7</th>
<th>PCS6</th>
<th>PCS5</th>
<th>PCS4</th>
<th>PCS3</th>
<th>PCS2</th>
<th>PCS1</th>
<th>PCS0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0  **PCS[15:0][1:0]**: SRD pending request clear input signal selection on event input x = truncate (n/2)
00: DMA channel 6 event selected as SRD domain pendclear source
01: DMA channel 7 event selected as SRD domain pendclear source
10: LPTIM2 out selected as SRD domain pendclear source
11: LPTIM3 out selected as SRD domain pendclear source
### 20.6.6 EXTI SRD pending clear selection register high (EXTI_SRDPCR1H)

Address offset: 0x14  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>PCS25[1:0]: SRD pending request clear input signal selection on Event input x = truncate (((n+32)/2))</td>
</tr>
<tr>
<td></td>
<td>00: DMA channel 6 event selected as SRD domain pendclear source</td>
</tr>
<tr>
<td></td>
<td>01: DMA channel 7 event selected as SRD domain pendclear source</td>
</tr>
<tr>
<td></td>
<td>10: LPTIM2 out selected as SRD domain pendclear source</td>
</tr>
<tr>
<td></td>
<td>11: LPTIM3 out selected as SRD domain pendclear source</td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>13</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>12</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>10</td>
<td>PCS21[1:0]: SRD pending request clear input signal selection on Event input x = truncate (((n+32)/2))</td>
</tr>
<tr>
<td></td>
<td>00: DMA channel 6 event selected as SRD domain pendclear source</td>
</tr>
<tr>
<td></td>
<td>01: DMA channel 7 event selected as SRD domain pendclear source</td>
</tr>
<tr>
<td></td>
<td>10: LPTIM2 out selected as SRD domain pendclear source</td>
</tr>
<tr>
<td></td>
<td>11: LPTIM3 out selected as SRD domain pendclear source</td>
</tr>
<tr>
<td>9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>6</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>4</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>3</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>2</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>1</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

### 20.6.7 EXTI rising trigger selection register (EXTI_RTSR2)

Address offset: 0x20  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>TR51[1:0]: SRD pending request clear input signal selection on Event input x = truncate (((n+32)/2))</td>
</tr>
<tr>
<td></td>
<td>00: DMA channel 6 event selected as SRD domain pendclear source</td>
</tr>
<tr>
<td></td>
<td>01: DMA channel 7 event selected as SRD domain pendclear source</td>
</tr>
<tr>
<td></td>
<td>10: LPTIM2 out selected as SRD domain pendclear source</td>
</tr>
<tr>
<td></td>
<td>11: LPTIM3 out selected as SRD domain pendclear source</td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>13</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>12</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>6</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>4</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>3</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>2</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>1</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
20.6.8 EXTI falling trigger selection register (EXTI_FTSR2)

Address offset: 0x24
Reset value: 0x0000 0000

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **TR51**: Rising trigger event configuration bit of configurable event input x+32 (x = 51)

0: Rising trigger disabled (for event and Interrupt) for input line
1: Rising trigger enabled (for event and Interrupt) for input line

*Note:* The configurable event inputs are edge triggered, no glitch must be generated on these inputs.

If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **TR49**: Rising trigger event configuration bit of configurable event input x+32 (x = 49)

0: Rising trigger disabled (for event and Interrupt) for input line
1: Rising trigger enabled (for event and Interrupt) for input line

*Note:* The configurable event inputs are edge triggered, no glitch must be generated on these inputs.

If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 16:0 Reserved, must be kept at reset value.

Bit 19 **TR51**: Falling trigger event configuration bit of configurable event input x+32 (x = 51)

0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line

*Note:* The configurable event inputs are edge triggered, no glitch must be generated on these inputs.

If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.
Bit 18 Reserved, must be kept at reset value.

Bit 17 **TR49**: Falling trigger event configuration bit of configurable event input x+32 (x = 49)
   0: Falling trigger disabled (for event and Interrupt) for input line
   1: Falling trigger enabled (for event and Interrupt) for input line

*Note:* The configurable event inputs are edge triggered, no glitch must be generated on these inputs.

If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 16:0 Reserved, must be kept at reset value.

### 20.6.9 EXTI software interrupt event register (EXTI_SWIER2)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **SWIER51**: Software interrupt on line x+32 (x = 51)
   This bit always returns 0 when read.
   0: Writing 0 has no effect.
   1: Writing a 1 to this bit will trigger an event on line x. This bit is automatically cleared by hardware.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **SWIER49**: Software interrupt on line x+32 (x = 49)
   This bit always returns 0 when read.
   0: Writing 0 has no effect.
   1: Writing a 1 to this bit will trigger an event on line x. This bit is automatically cleared by hardware.

Bits 16:0 Reserved, must be kept at reset value.
### 20.6.10 EXTI SRD pending mask register (EXTI_SRDPMR2)

Address offset: 0x2C  
Reset value: 0x0000 0000

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

Bits 31:20  Reserved, must be kept at reset value.

Bits 19:16 **MR[51:48]**: SRD pending mask on event input x+32 (x = 51 to 48)
- 0: SRD pending request from line x+32 is masked. Writing this bit to 0 also clears the SRD pending request.
- 1: SRD pending request from line x+32 is unmasked. When triggered, the SRD domain pending signal keeps SRD domain wakeup active until cleared.

Bits 15:10  Reserved, must be kept at reset value.

Bit 9 **MR41**: SRD pending mask on event input x+32 (x = 41)
- 0: SRD pending request from Line x+32 is masked. Writing this bit to 0 also clears the SRD Pending request.
- 1: SRD pending request from Line x+32 is unmasked. When triggered, the SRD domain pending signal keeps SRD domain wakeup active until cleared.

Bits 8:4  Reserved, must be kept at reset value.

Bits 3:2 **MR[35:34]**: SRD Pending Mask on Event input x+32 (x = 35 to 34)
- 0: SRD pending request from Line x+32 is masked. Writing this bit to 0 also clears the SRD Pending request.
- 1: SRD pending request from Line x+32 is unmasked. When triggered, the SRD domain pending request keeps SRD domain wakeup active until cleared.

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

### 20.6.11 EXTI SRD pending clear selection register low (EXTI_SRDPCCR2L)

Address offset: 0x30  
Reset value: 0x0000 0000

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

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

Bits 31:20  Reserved, must be kept at reset value.

Bits 19:16 **PCS41**: SRD pending clear on event input x+32 (x = 41)
- 0: SRD pending clear from Line x+32 is masked. Writing this bit to 0 also clears the SRD pending clear.
- 1: SRD pending clear from Line x+32 is unmasked. When triggered, the SRD domain pending clear signal keeps SRD domain wakeup active until cleared.

Bits 15:10  Reserved, must be kept at reset value.

Bits 9:4 **PCS35**: SRD Pending Clear Mask on Event input x+32 (x = 35 to 34)
- 0: SRD pending clear from Line x+32 is masked. Writing this bit to 0 also clears the SRD pending clear.
- 1: SRD pending clear from Line x+32 is unmasked. When triggered, the SRD domain pending clear signal keeps SRD domain wakeup active until cleared.

Bits 3:0  Reserved, must be kept at reset value.
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:18  PCS41[1:0]: SRD Pending request clear input signal selection on Event input x = truncate \((n+64)/2\)

  00: DMA ch6 event selected as SRD domain pendclear source
  01: DMA ch7 event selected as SRD domain pendclear source
  10: LPTIM2 out selected as SRD domain pendclear source
  11: LPTIM3 out selected as SRD domain pendclear source

Bits 17:8  Reserved, must be kept at reset value.

Bits 7:4  PCS[35:34][1:0]: SRD Pending request clear input signal selection on Event input x= truncate\n\((n+64)/2\)

  00: DMA ch6 event selected as SRD domain pendclear source
  01: DMA ch7 event selected as SRD domain pendclear source
  10: LPTIM2 out selected as SRD domain pendclear source
  11: LPTIM3 out selected as SRD domain pendclear source

Bits 3:0  Reserved, must be kept at reset value.

20.6.12  EXTI SRD pending clear selection register high (EXTI_SRDPCR2H)

Address offset: 0x34

Reset value: 0x0000 0000

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

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

Bits 7:0  PCS[51:48][1:0]: SRD Pending request clear input signal selection on Event input x= truncate\n\((n+96)/2\)

  00: DMA ch6 event selected as SRD domain pendclear source
  01: DMA ch7 event selected as SRD domain pendclear source
  10: LPTIM2 out selected as SRD domain pendclear source
  11: LPTIM3 out selected as SRD domain pendclear source
### 20.6.13 EXTI rising trigger selection register (EXTI_RTSR3)

**Address offset:** 0x40  
**Reset value:** 0x0000 0000

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

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

Bit 21 **TR85:** Rising trigger event configuration bit of configurable event input x+64 (x = 85)  
0: Rising trigger disabled (for event and Interrupt) for input line  
1: Rising trigger enabled (for event and Interrupt) for input line  

*Note:* The configurable event inputs are edge triggered, no glitch must be generated on these inputs.  
If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.  
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TR82:** Rising trigger event configuration bit of configurable event input x+64 (x = 82)  
0: Rising trigger disabled (for event and Interrupt) for input line  
1: Rising trigger enabled (for event and Interrupt) for input line  

*Note:* The configurable event inputs are edge triggered, no glitch must be generated on these inputs.  
If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.  
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 17:0 Reserved, must be kept at reset value.

### 20.6.14 EXTI falling trigger selection register (EXTI_FTSR3)

**Address offset:** 0x44  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TR85</td>
<td>TR82</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Note:** The configurable event inputs are edge triggered, no glitch must be generated on these inputs.  
If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.  
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.
Extended interrupt and event controller (EXTI) RM0455

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

Bit 21 **TR[85]**: Falling trigger event configuration bit of configurable event input x+64 (x = 85)
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line

*Note:* The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TR82**: Falling trigger event configuration bit of configurable event input x+64 (x = 82)
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line

*Note:* The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 17:0 Reserved, must be kept at reset value.

### 20.6.15 EXTI software interrupt event register (EXTI_SWIER3)

**Address offset:** 0x48

**Reset value:** 0x0000 0000

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

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

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

Bit 21 **SWIER[85]**: Software interrupt on line x+64 (x = 85)
This bitfield always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit will trigger an event on line x. This bit is automatically cleared by hardware.
Bits 20:19  Reserved, must be kept at reset value.

Bit 18 **SWIER82**: Software interrupt on line x+64 (x = 82)
This bitfield always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit will trigger an event on line x. This bit is automatically cleared by hardware.

Bits 17:0  Reserved, must be kept at reset value.

### 20.6.16 EXTI SRD pending mask register (EXTI_SRDMR3)

Address offset: 0x4C
Reset value: 0x0000 0000

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

Bits 31:25  Reserved, must be kept at reset value.

Bit 24 **MR88**: SRD pending mask on event input x+64 (x = 88)
0: SRD pending request from Line x+64 is masked. Writing this bit to 0 also clears the SRD pending request.
1: SRD Pending request from Line x+64 is unmasked. When triggered, the SRD domain pending signal keeps SRD domain wakeup active until cleared.

Bits 23:0  Reserved, must be kept at reset value.

### 20.6.17 EXTI SRD pending clear selection register high (EXTI_SRDPCHR3H)

Address offset: 0x54
Reset value: 0x0000 0000

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

Bits 31:25  Reserved, must be kept at reset value.

Bit 24 **PCS88**: SRD pending clear selection on event input x+64 (x = 88)
0: SRD pending request from Line x+64 is masked. Writing this bit to 0 also clears the SRD pending signal.
1: SRD pending request from Line x+64 is unmasked. When triggered, the SRD domain pending signal keeps SRD domain wakeup active until cleared.

Bits 23:0  Reserved, must be kept at reset value.
### EXTI interrupt mask register (EXTI_CPUIMR1)

**Address offset:** 0x80  
**Reset value:** 0xFFC0 0000

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

**Bits 31:18** Reserved, must be kept at reset value.

**Bits 17:16** **PCS88[1:0]:** SRD pending request clear input signal selection on event input x= truncate \((n+160)/2\)

- 00: DMA channel 6 event selected as SRD domain pendclear source
- 01: DMA channel 7 event selected as SRD domain pendclear source
- 10: LPTIM2 out selected as SRD domain pendclear source
- 11: LPTIM3 out selected as SRD domain pendclear source

**Bits 15:0** Reserved, must be kept at reset value.

### EXTI event mask register (EXTI_CPUEMR1)

**Address offset:** 0x84  
**Reset value:** 0x0000 0000

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

**Bits 31:22** **MR[31:22]:** CPU interrupt mask on direct event input x (x = 31 to 22)

- 0: Interrupt request from Line x is masked
- 1: Interrupt request from Line x is unmasked

**Note:** The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.

**Bits 21:0** **MR[21:0]:** CPU interrupt mask on configurable event input x (x = 21 to 0)

- 0: Interrupt request from Line x is masked
- 1: Interrupt request from Line x is unmasked

**Note:** The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.
### 20.6.20 EXTI pending register (EXTI_CPUPR1)

Address offset: 0x88  
Reset value: undefined

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PR15</td>
<td>PR14</td>
<td>PR13</td>
<td>PR12</td>
<td>PR11</td>
<td>PR10</td>
<td>PR9</td>
<td>PR8</td>
<td>PR7</td>
<td>PR6</td>
<td>PR5</td>
<td>PR4</td>
<td>PR3</td>
<td>PR2</td>
<td>PR1</td>
<td>PR0</td>
</tr>
<tr>
<td>rc1</td>
<td>rc1</td>
<td>rc1</td>
<td>rc1</td>
<td>rc1</td>
<td>rc1</td>
<td>rc1</td>
<td>rc1</td>
<td>rc1</td>
<td>rc1</td>
<td>rc1</td>
<td>rc1</td>
<td>rc1</td>
<td>rc1</td>
<td>rc1</td>
<td></td>
</tr>
</tbody>
</table>

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

**Bits 21:0** **PR[21:0]:** Configurable event inputs x Pending bit (x = 21 to 0)
- 0: No trigger request occurred
- 1: selected trigger request occurred

This bit is set when the selected edge event is detected on the external interrupt line. This bit is cleared by writing 1 to the bit or by changing the edge detector sensitivity.

### 20.6.21 EXTI interrupt mask register (EXTI_CPUIMR2)

Address offset: 0x90  
Reset value: 0xFFF5 FFFF

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

**Bits 31:20** **MR[63:52]:** CPU interrupt mask on direct event input x+32 (x = 63 to 52)
- 0: Interrupt request from Line x is masked
- 1: Interrupt request from Line x is unmasked

**Note:** The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.

**Bit 19** **MR51:** CPU interrupt mask on configurable event input x+32 (x = 51)
- 0: Interrupt request from Line x is masked
- 1: Interrupt request from Line x is unmasked

**Note:** The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

**Bit 18** **MR50:** CPU Interrupt mask on direct Event input x+32 (x = 50)
- 0: Interrupt request from Line x is masked
- 1: Interrupt request from Line x is unmasked

**Note:** The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.
Bit 17 **MR49**: CPU interrupt mask on configurable event input \(x+32\) (\(x = 49\))
- 0: Interrupt request from Line \(x\) is masked
- 1: Interrupt request from Line \(x\) is unmasked

*Note: The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.*

Bits 16:15 **MR[48:47]**: CPU interrupt mask on direct event input \(x+32\) (\(x = 47\) to 46)
- 0: Interrupt request from Line \(x\) is masked
- 1: Interrupt request from Line \(x\) is unmasked

*Note: The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.*

Bits 14:12 Reserved, must be kept at reset value.

*Note: The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.*

Bits 11:0 **MR[43:32]**: CPU interrupt mask on direct event input \(x+32\) (\(x = 43\) to 32)
- 0: Interrupt request from Line \(x\) is masked
- 1: Interrupt request from Line \(x\) is unmasked

*Note: The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.*

### 20.6.22 EXTI event mask register (EXTI_CPUEMR2)

**Address offset**: 0x94

**Reset value**: 0x0000 0000

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

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

Bits 31:15 **MR[63:47]** CPU event mask on event input \(x+32\) (\(x = 63\) to 47)
- 0: Event request from Line \(x\) is masked
- 1: Event request from Line \(x\) is unmasked

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:0 **MR[43:32]**: CPU event mask on event input \(x+32\) (\(x = 43\) to 32)
- 0: Event request from Line \(x\) is masked
- 1: Event request from Line \(x\) is unmasked
20.6.23 EXTI pending register (EXTI_CPUPR2)

Address offset: 0x98
Reset value: undefined

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

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **PR51**: Configurable event inputs x+32 pending bit (x = 51)
- 0: No trigger request occurred
- 1: Selected trigger request occurred
  This bit is set when the selected edge event is detected on the external interrupt line. This bit is cleared by writing a 1 to the bit or by changing the edge detector sensitivity.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PR49**: Configurable event inputs x+32 pending bit (x = 49)
- 0: No trigger request occurred
- 1: Selected trigger request occurred
  This bit is set when the selected edge event is detected on the external interrupt line. This bit is cleared by writing a 1 to the bit or by changing the edge detector sensitivity.

Bits 16:0 Reserved, must be kept at reset value.

20.6.24 EXTI interrupt mask register (EXTI_CPUIMR3)

Address offset: 0xA0
Reset value: 0x018B FFFF

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

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:23 **MR[88:87]**: CPU interrupt mask on direct event input x+64 (x =88, 87)
- 0: Interrupt request from Line x is masked
- 1: Interrupt request from Line x is unmasked
  
  *Note:* The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.

Bit 22 Reserved, must be kept at reset value.
Bit 21  **MR[85]**: CPU interrupt mask on configurable event input x+64 (x = 85)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

*Note:* The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

Bits 20:19  Reserved, must be kept at reset value.

*Note:* The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.

Bit 18  **MR82**: CPU interrupt mask on configurable event input x+64 (x = 82)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

*Note:* The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

Bit 17  Reserved, must be kept at reset value.

*Note:* The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.

Bit 16  **MR[80]**: CPU interrupt mask on direct event input x+64 (x = 80)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

*Note:* The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.

Bits 15:14  Reserved, must be kept at reset value.

*Note:* The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.

Bit 13  **MR[77]**: CPU interrupt mask on direct event input x+64 (x = 77)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

*Note:* The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.

Bits 12:11  Reserved, must be kept at reset value.

*Note:* The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.

Bits 12:0  **MR[74:54]**: CPU interrupt mask on direct event input x+64 (x = 74 to 54)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

*Note:* The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.
**20.6.25 EXTI event mask register (EXTI_CPUEMR3)**

Address offset: 0xA4  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:25</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>24:23</td>
<td><strong>MR[88:87]</strong>: CPU event mask on event input x+64 (x = 88, 87)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Event request from Line x is masked</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1: Event request from Line x is unmasked</td>
<td>1</td>
</tr>
<tr>
<td>Bit 22</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>Bit 21</td>
<td><strong>MR85</strong>: CPU event mask on event input x+64 (x = 85)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Event request from Line x is masked</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1: Event request from Line x is unmasked</td>
<td>1</td>
</tr>
<tr>
<td>20:19</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>Bit 18</td>
<td><strong>MR82</strong>: CPU event mask on event input x+64 (x = 82)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Event request from Line x is masked</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1: Event request from Line x is unmasked</td>
<td>1</td>
</tr>
<tr>
<td>Bit 17</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>Bit 16</td>
<td><strong>MR[80]</strong>: CPU event mask on event input x+64 (x = 80)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Event request from Line x is masked</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1: Event request from Line x is unmasked</td>
<td>1</td>
</tr>
<tr>
<td>15:14</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>Bit 13</td>
<td><strong>MR77</strong>: CPU event mask on event input x+64 (x = 77)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Event request from Line x is masked</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1: Event request from Line x is unmasked</td>
<td>1</td>
</tr>
<tr>
<td>12:11</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>10:0</td>
<td><strong>MR[74:64]</strong>: CPU event mask on event input x+64 (x = 74 to 64)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Event request from Line x is masked</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1: Event request from Line x is unmasked</td>
<td>1</td>
</tr>
</tbody>
</table>

Bits 31:25: Reserved, must be kept at reset value.

Bits 24:23: **MR[88:87]**: CPU event mask on event input x+64 (x = 88, 87)

- 0: Event request from Line x is masked
- 1: Event request from Line x is unmasked

Bit 22: Reserved, must be kept at reset value.

Bit 21: **MR85**: CPU event mask on event input x+64 (x = 85)

- 0: Event request from Line x is masked
- 1: Event request from Line x is unmasked

Bits 20:19: Reserved, must be kept at reset value.

Bit 18: **MR82**: CPU event mask on event input x+64 (x = 82)

- 0: Event request from Line x is masked
- 1: Event request from Line x is unmasked

Bit 17: Reserved, must be kept at reset value.

Bit 16: **MR[80]**: CPU event mask on event input x+64 (x = 80)

- 0: Event request from Line x is masked
- 1: Event request from Line x is unmasked

Bits 15:14: Reserved, must be kept at reset value.

Bit 13: **MR77**: CPU event mask on event input x+64 (x = 77)

- 0: Event request from Line x is masked
- 1: Event request from Line x is unmasked

Bits 12:11: Reserved, must be kept at reset value.

Bits 10:0: **MR[74:64]**: CPU event mask on event input x+64 (x = 74 to 64)

- 0: Event request from Line x is masked
- 1: Event request from Line x is unmasked
Extended interrupt and event controller (EXTI)  RM0455

20.6.26 EXTI pending register (EXTI_CPUPR3)

Address offset: 0xA8
Reset value: undefined

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

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

Bit 21 **PR[85]**: Configurable event inputs x+64 pending bit (x = 85)
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event is detected on the external interrupt line. This bit is cleared by writing 1 to the bit or by changing the edge detector sensitivity.

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **PR82**: Configurable event inputs x+64 pending bit (x = 82)
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event is detected on the external interrupt line. This bit is cleared by writing 1 to the bit or by changing the edge detector sensitivity.

Bits 17:0 Reserved, must be kept at reset value.

20.6.27 EXTI register map

The following table gives the EXTI register map and the reset values.

Table 128. Asynchronous interrupt/event controller register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>EXTI_RTSR1</td>
<td>TR[21:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>U U U U U U U U U U U U U U U U U U U U U U</td>
</tr>
<tr>
<td>0x04</td>
<td>EXTI_FTSR1</td>
<td>TR[21:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x08</td>
<td>EXTI_SWIER1</td>
<td>SWIER[21:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>U U U U U U U U U U U U U U U U U U U U U U</td>
</tr>
<tr>
<td>0x0C</td>
<td>EXTI_SRDPMR1</td>
<td>MR[21:19]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>U U U U U U U U U U U U U U U U U U U U U U</td>
</tr>
</tbody>
</table>

752/2974 RM0455 Rev 3
### Table 128. Asynchronous interrupt/event controller register map and reset values (continued)

|--------|----------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|        |
| 0x10   | EXTI_SRDPCTR1L |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value    | U      | U      | U      | U      | U      | U      | U      | U      | U      | U      | U      | U      | U      | U      | U      | U      | U      |        |        |
| 0x14   | EXTI_SRDPCTR1H |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value    | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |        |        |
| 0x20   | EXTI_RTSR2     |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value    |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x24   | EXTI_FTSR2     |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value    |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x28   | EXTI_SWIER2    |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value    |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x2C   | EXTI_SRDPMR2   |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value    | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |        |        |
| 0x30   | EXTI_SRDPCR2L  |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value    | 0      | 0      |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x34   | EXTI_SRDPCR2H  |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value    | 0      | 0      |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x40   | EXTI_RTSR3     |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value    |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x44   | EXTI_FTSR3     |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value    |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x48   | EXTI_SWIER3    |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value    |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x4C   | EXTI_SRDPMR3   |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value    | c      | c      | c      | c      | c      | c      | c      | c      | c      | c      | c      | c      | c      | c      | c      | c      | c      | c      | c      |
| 0x50   | Reserved       |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x54   | EXTI_SRDPCR3H  |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value    | 0      | 0      |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x58-  | Reserved       |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x7C   |                |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
### Table 128. Asynchronous interrupt/event controller register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset name</th>
<th>Reset value</th>
<th>Offset name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x80</td>
<td>EXTI_CPUIMR1</td>
<td>MR[31:22]</td>
<td>1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>MR[21:0]</td>
</tr>
<tr>
<td>0x84</td>
<td>EXTI_CPUEMR1</td>
<td>MR[31:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0x88</td>
<td>EXTI_CPUPR1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x90</td>
<td>EXTI_CPUIMR2</td>
<td>MR[63:47]</td>
<td>1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
<td>MR[43:32]</td>
</tr>
<tr>
<td>0x94</td>
<td>EXTI_CPUEMR2</td>
<td>MR[63:47]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>MR[43:32]</td>
</tr>
<tr>
<td>0x98</td>
<td>EXTI_CPUPR2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xA0</td>
<td>EXTI_CPUIMR3</td>
<td>MR[88]</td>
<td>1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>MR[74:64]</td>
</tr>
<tr>
<td>0xA4</td>
<td>EXTI_CPUEMR3</td>
<td>MR[88]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>MR[74:64]</td>
</tr>
<tr>
<td>0xAB</td>
<td>EXTI_CPUPR3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xAC</td>
<td>Reserved</td>
<td>0 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
21 Chrom-GRC™ (GFXMMU)

21.1 Introduction
The graphic MMU is a graphical oriented memory management unit aimed to:
- Optimize memory usage according to the display shape
- Cache linear accesses to the frame buffer
- Prefetch data

21.2 Chrom-GRC™ main features
- Fully programmable display shape to physically store only the visible pixel
- Up to 4 virtual buffer
- Each virtual buffer have 3072 or 4096 bytes per line and 1024 lines
- Each virtual buffer can be physically mapped to any system memory
- Optional cache for linear accesses
- Cache can be locked to a virtual buffer
- Cache prefetch mechanism for linear accesses anticipation
- Interrupt in case of buffer overflow (1 per buffer)
- Interrupt in case of memory transfer error
21.3 Chrom-GRC™ functional and architectural description

The graphic MMU is composed of two main block:
- The MMU, responsible of address resolution
- The cache, responsible of data/address caching, prefetching and packing/unpacking

**Figure 99. Graphic MMU block diagram**

21.3.1 Virtual memory

The graphic MMU provides a virtual memory space seen by all the system masters. This virtual memory space is divided into four virtual buffers.

**Virtual buffer**

A virtual buffer is seen by any system master as a continuous memory space representing a virtual frame buffer of 1024 lines.

Each line is divided into 192 or 256 16-byte blocks depending on the 192BM bit of the graphic MMU configuration register (GFXMMU_CR).

Depending on the display shape and size, only the necessary blocks will be mapped to a physical memory location. This mapping is done programming the LUT entry for each line:
- The enable of the line
- The number of the first “visible” block
- The number of the last “visible” block
- The address offset of the line within the physical buffer

The “visible” blocks can be arranged in the physical buffer in a continuous way programming the address offset of each line.

The LUT is common to all the buffers i.e. all the buffers have the same “shape”.

MSv41699V1
Virtual buffer overview

For a frame buffer coded in 32bpp or 16bpp, the virtual buffer can be configured to have 192 or 256 blocks. This will result in a virtual frame buffer of 768 x 1024 or 1024 x 1024 pixels for 32bpp and 1536 x 1024 or 2048 x 1024 for 16bpp.

For a frame buffer coded in 24bpp, the virtual buffer shall be configured to have 192 blocks to have an integer number of pixels per line. This will result in a virtual frame buffer of 1024 x 1024 pixels for 24bpp.

Each buffer can be physically mapped anywhere in the physical memory thanks to:

- The physical buffer base address (PBBA) field of the graphic MMU buffer x configuration register (GFXMMU_BxCR). It configures the physical location of the 8 MByte area where the buffer is mapped.
- The physical buffer location respective to the physical buffer base address is defined by the physical buffer offset (PBO) field of the graphic MMU buffer x configuration register (GFXMMU_BxCR).
Virtual buffer application use case

As the physical locations are independently configurable, the four virtual buffers can be physically mapped to non continuous locations. This would allow for example to have the four buffers mapped on to four different SDRAM banks and avoid extra precharge cycles accessing the SDRAM.

As a consequence, one buffer shall be used by the CPU/ChromART® for frame buffer calculation while an other one shall be used by the LTDC.

The two remaining buffers can be used as a graphical library requiring extra drawing buffers.

21.3.2 MMU architecture

The MMU block is responsible of the address resolution. It receives the 24-bit address and returns the physical 23-bit address and a valid signals to indicate the address is physically mapped or not. The MMU also checks overflow of a area boundary.
The MMU LUT is implemented as a 1024 x 35-bit RAM

**Figure 102. MMU block diagram**

**Line block decoder**

The line block decoder is generating the block number and the line number according the address.

**Look up RAM**

The look up RAM is a 1024 x 35-bit RAM with the following fields:
- 1-bit line enable
- 8-bit first valid block
- 8-bit last valid block
- 18-bit for line offset

As the RAM is bigger than a word, each entry is split into two words on the memory map. The write access are done in two steps:

1. Write the first word with enable/first valid block/last valid block in the GFXMMU_LUTxL memory location (internally buffered)
2. Write the second word with line offset in the GFXMMU_LUTxH memory location (effective write into the memory together with the internally buffered value)

A write in the LUT can happen any time but it can lead to inconsistencies if a master is using the MMU at the same time. As the CPU has the priority during LUT programming, this may slow down MMU calculation.

There is no restriction during read operations, but this may slow down CPU as the MMU has the priority on LUT accesses.

**Block validation/comparator**

This block is checking is the block is valid.
A block is considered as valid i.e. physically mapped when
- Line is enable
- The block number is greater or equal to the first valid block
- The block number is lower or equal to the last valid block

When the block is valid, the physical address generated is considered as correct.

If the result of the MMU evaluation is not valid, the write operations are ignored, and read operations return the default 32-bit value stored in the default value (DV) field of the graphic MMU default value register (GFXMMU_DVR).

**Block offset address calculation within the buffer**

The block number is added to the line offset to get the offset of the block within the physical buffer.

As a consequence, the line offset stored in the LUT is given by the following formula:

\[
\text{Line offset} = [(\text{Number of visible blocks already used}) - (1^{st} \text{ visible block})] \times \text{block size}
\]

with:
- The maximum value for the line offset is when all the block of all the line are used. As the consequence the line offset for the last line can be maximum:
  \[1023 \times 256 \times 6 = 0\times F00x\]
- The minimum value for the line offset is when the last block of the first line is the first valid block: \[-255\times 16 = - 0xFFx \text{ i.e } 0\times F01x\]

As the consequence the full range of the line offset entry of the LUT is used.

Carry is not taken into account as this stage to be able to perform negative offset calculations (values from 0x3F:F01x to 0x3F:FFFx)

As the block offset is within a 4 MByte buffer, the address generated is 22-bit wide.

**Block offset address calculation**

Once the offset of the block within the buffer as been calculated, this value is added to the offset of the block respective to the physical buffer base address.

The offset of the blocks are defined in registers as shown

![Figure 103. Block validation/comparator implementation](image)

The resulting address and the buffer offset address shall be on 23-bit.

The carry is taken into account to trigger address overflow. The carry is propagated to the graphic MMU status register (GFXMMU_SR) to set the buffer x overflow flag (BxOF).
Example of calculation

We are considering the following configuration for virtual buffer 0

- First visible block of line 0: block 7
- Number of visible block in line 0: 10
- First visible block of line 1: block 6
- Number of visible block in line 1: 12
- Address of the physical buffer: 0xC020:0000

The configuration shall be:

- The base address of the physical buffer 0: 0xC000:0000
- The offset of buffer 0: 0x20:0000
- First visible block of line 0: block 7
- Last visible block of line 0: block 16
- Block 0 offset of line 0: \((0 - 7) \times 0x10 = -0x70 = 0x3F:FF90\)
- First visible block of line 1: block 6
- Last visible block of line 1: block 17
- Block 0 offset of line 1: \((10 - 6) \times 0x10 = (0xA - 0x6) \times 0x10 = 0x40\)

As a consequence:

- the physical address of block 7 of line 0 is:
  \(0xC000:0000 + 0x20:0000 + (0x3F:FF90 + 0x70 \text{ without carry}) = 0xC020:0000\)

- the physical address of block 16 of line 0 is:
  \(0xC000:0000 + 0x20:0000 + (0x3F:FF90 + 0x100 \text{ without carry}) = 0xC020:0090\)

- the physical address of block 6 of line 1 is:
  \(0xC000:0000 + 0x20:0000 + (0x40 + 0x60 \text{ without carry}) = 0xC020:00A0\)

- the physical address of block 17 of line 1 is:
  \(0xC000:0000 + 0x20:0000 + (0x40 + 0x110 \text{ without carry}) = 0xC020:0150\)

### 21.3.3 Cache and prefetch mechanism

The GFXMMU integrated cache target internal or external RAM devices storing the graphical frame buffer.

**Master accessing the graphic MMU**

Several masters are supposed to access memories through the graphic MMU

- When the CPU is accessing the framebuffer, it’s because it’s performing a read/modify/write of a single pixel in the frame buffer. As a consequence the granularity of the R/M/W is less or equal to a word and a data cache is necessary. As the frame buffer is scanned linearly, it’s realistic to says that the next data that will be requested by the CPU within a buffer is the next pixel. A prefetch mechanism can anticipate this efficiently.

- When ChromART® or the LTDC are accessing the framebuffer, they are generating long accesses up to 128 Byte. A cache may not be necessary in this case as successive operation will be done only once.
As a consequence the cache is aimed for CPU and shall not be used together with DMA2D or LTDC.

**Cache enabling**

The cache is enabled setting the CE (cache enable) bit of the graphic MMU control register (GFXMMU_CR).

**Cache lock mechanism**

It’s possible to lock the cache by setting the cache lock bit (CL) and configuring the cache lock buffer (CLB) field of the graphic MMU configuration register (GFXMMU_CR).

When the cache is locked on a given buffer, all the other buffer can not be cached.

An operation on the locked buffered can be cachable according to attribute of the master request, or can be always cachable setting the force caching (FC) bit of the graphic MMU configuration register (GFXMMU_CR). The force caching mechanism is only available when the cache is locked to a buffer. The force caching (FC) bit of the graphic MMU configuration register (GFXMMU_CR) is automatically reset when the cache lock (CL) bit of the graphic MMU configuration register (GFXMMU_CR) is reset.

**Cache line size**

A cache line size is 32-byte.

**Number of cache lines**

The number of line is reduced as much as possible taking into account that CPU is performing linear accesses.

As a consequence 3 lines are needed:

- 1 line for the current access
- 1 line for the previous access (as we can have pixels split into 2 16-byte blocks)
- 1 line for the prefetched access

**Prefetch mechanism**

Because the CPU will most of the time perform linear accesses to the frame buffer, a prefetch mechanism is provided in order to automatically retrieve the next cache line from the memory.

The prefetch mechanism can be disabled setting the prefetch disable (PD) bit of the graphic MMU configuration register (GFXMMU_CR).

When the prefetch mechanism is disabled, only the TAG of the line dedicated for prefetching is updated but the data are retrieved from the memory.

**Cache maintenance operation**

When the cache is working in non force caching mode i.e. the force caching (FC) bit of the graphic MMU configuration register (GFXMMU_CR) is reset, then line eviction is done:

- Each time a non cachable access is performed on the buffer having data cached
- Each time a miss occurs

As a consequence the cache maintenance operation can be naturally automatic.
Nevertheless, it’s possible to force a flush if the cache setting the force flush (FF) bit of the graphic MMU cache control register (GFXMMU_CCR). When flushing the cache, all the dirty entries are sent to the write buffer, and all the dirty bit of the TAGs are reset. But the entries are not invalidate.

It’s also possible to invalidate the cache entries setting the force invalidate (FI) bit of the graphic MMU cache control register (GFXMMU_CCR). This will not send the dirty entries to the write buffer (modification done in the cache are lost). This will just reset the ID field the TAGs (unused state)

Setting the two bits force flush (FF) and force invalidate (FI) of the graphic MMU cache control register (GFXMMU_CCR) will:
- flush the cache, all the dirty entries are sent to the write buffer
- reset the dirty bit of the TAGs
- reset the ID field of the TAGs

When any of these two operations is done while the flash is having transaction already in the write buffer, the write buffer continue its operations.

The force flush (FF) and force invalidate (FI) are reset automatically when all the operations are finished (write buffer empty) creating a synchronization barrier.
21.4 Graphic MMU interrupts

An interrupt can be produced on the following events:
- Buffer 0 overflow
- Buffer 1 overflow
- Buffer 2 overflow
- Buffer 3 overflow
- AHB master error

Separate interrupt enable bits are available for flexibility.

Table 129. Graphic MMU interrupt requests

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer 0 overflow</td>
<td>B0OF</td>
<td>B0OIE</td>
</tr>
<tr>
<td>Buffer 1 overflow</td>
<td>B1OF</td>
<td>B1OIE</td>
</tr>
<tr>
<td>Buffer 2 overflow</td>
<td>B2OF</td>
<td>B2OIE</td>
</tr>
<tr>
<td>Buffer 3 overflow</td>
<td>B3OF</td>
<td>B3OIE</td>
</tr>
<tr>
<td>AHB master error</td>
<td>AMEF</td>
<td>AMEIE</td>
</tr>
</tbody>
</table>
21.5 Graphic MMU registers

21.5.1 Graphic MMU configuration register (GFXMMU_CR)

Address offset: 0x0000
Reset value: 0x0000 0000

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

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

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **OB**: Outer bufferability
This bit configures the bufferability of an access generated by the graphic MMU cache.
0: No bufferable
1: Bufferable

Bit 16 **OC**: Outer cachability
This bit configures the cachability of an access generated by the graphic MMU cache.
0: No cachable
1: Cachable

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PD**: Prefetch disable
This bit disables the prefetch of the cache.
0: Prefetch enable
1: Prefetch disable

Bit 11 **FC**: Force caching
This bit forces the caching into the cache regardless of the MPU attributes.
The cache must be enable (CE bit set).
0: Caching not forced
1: Caching forced

Bits 10:9 **CLB[1:0]**: Cache lock buffer
This field selects the buffer on which the cache is locked.
00: Cache locked on buffer 0
01: Cache locked on buffer 1
10: Cache locked on buffer 2
11: Cache locked on buffer 3

Bit 8 **CL**: Cache lock
This bit locks the cache onto the buffer defined in the CLB field.
0: Cache not locked
1: Cache locked to a buffer

Bit 7 **CE**: Cache enable
This bit enables the cache unit.
0: Cache disable
1: Cache enable
Bit 6 **192BM**: 192 Block mode
   This bit defines the number of blocks per line
   0: 256 blocks per line
   1: 192 blocks per line

Bit 5 Reserved, must be kept at reset value.

Bit 4 **AMEIE**: AHB master error interrupt enable
   This bit enables the AHB master error interrupt.
   0: Interrupt disable
   1: Interrupt enabled

Bit 3 **B3OIE**: Buffer 3 overflow interrupt enable
   This bit enables the buffer 3 overflow interrupt.
   0: Interrupt disable
   1: Interrupt enabled

Bit 2 **B2OIE**: Buffer 2 overflow interrupt enable
   This bit enables the buffer 2 overflow interrupt.
   0: Interrupt disable
   1: Interrupt enabled

Bit 1 **B1OIE**: Buffer 1 overflow interrupt enable
   This bit enables the buffer 1 overflow interrupt.
   0: Interrupt disable
   1: Interrupt enabled

Bit 0 **B0OIE**: Buffer 0 overflow interrupt enable
   This bit enables the buffer 0 overflow interrupt.
   0: Interrupt disable
   1: Interrupt enabled

### 21.5.2 Graphic MMU status register (GFXMMU_SR)

Address offset: 0x0004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>AMEF</td>
<td>B3OF</td>
<td>B2OF</td>
<td>B1OF</td>
<td>B0OF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **AMEF**: AHB master error flag
   This bit is set when an AHB error happens during a transaction. It is cleared by writing 1 to CAMEF.

Bit 3 **B3OF**: Buffer 3 overflow flag
   This bit is set when an overflow occurs during the offset calculation of the buffer 3. It is cleared by writing 1 to CB3OF.
Bit 2 \textbf{B2OF}: Buffer 2 overflow flag
This bit is set when an overflow occurs during the offset calculation of the buffer 2. It is cleared by writing 1 to CB2OF.

Bit 1 \textbf{B1OF}: Buffer 1 overflow flag
This bit is set when an overflow occurs during the offset calculation of the buffer 1. It is cleared by writing 1 to CB1OF.

Bit 0 \textbf{B0OF}: Buffer 0 overflow flag
This bit is set when an overflow occurs during the offset calculation of the buffer 0. It is cleared by writing 1 to CB0OF.

21.5.3 Graphic MMU flag clear register (GFXMMU_FCR)
Address offset: 0x0008
Reset value: 0x0000 0000

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

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 \textbf{CAMEF}: Clear AHB master error flag
Writing 1 clears the AHB master error flag in the GFXMMU_SR register.

Bit 3 \textbf{CB3OF}: Clear buffer 3 overflow flag
Writing 1 clears the buffer 3 overflow flag in the GFXMMU_SR register.

Bit 2 \textbf{CB2OF}: Clear buffer 2 overflow flag
Writing 1 clears the buffer 2 overflow flag in the GFXMMU SR register.

Bit 1 \textbf{CB1OF}: Clear buffer 1 overflow flag
Writing 1 clears the buffer 1 overflow flag in the GFXMMU_SR register.

Bit 0 \textbf{CB0OF}: Clear buffer 0 overflow flag
Writing 1 clears the buffer 0 overflow flag in the GFXMMU_SR register.

21.5.4 Graphic MMU cache control register (GFXMMU_CCR)
Address offset: 0x000C
Reset value: 0x0000 0000

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

|    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 31 | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
| FI | FF  | rs  | rs  |     |     |     |     |     |     |     |     |     |     |     |     |

STI
21.5.5 Graphic MMU default value register (GFXMMU_DVR)

Address offset: 0x0010
Reset value: 0x0000 0000

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

| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|-------------------------------|-------------------|
| DV[15:0]                      |                   |
| rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw |

Bits 31:0 **DV[31:0]: Default value**
This field indicates the default 32-bit value which is returned when a master accesses a virtual memory location not physically mapped.

21.5.6 Graphic MMU buffer 0 configuration register (GFXMMU_B0CR)

Address offset: 0x0020
Reset value: 0x0000 0000

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

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

| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|-------------------------------|-------------------|
| PBO[15:4]                    |                   |
| rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw |

Bits 31:23 **PBB[31:23]: Physical buffer base address**
Base address MSB of the physical buffer.

Bits 22:4 **PBO[22:4]: Physical buffer offset**
Offset of the physical buffer.

Bits 3:0 Reserved, must be kept at reset value.
### 21.5.7 Graphic MMU buffer 1 configuration register (GFXMMU_B1CR)

Address offset: 0x0024  
Reset value: 0x0000 0000

| Bits 31:23 | PBBA[31:23]: Physical buffer base address  
| Base address MSB of the physical buffer. |
| Bits 22:4 | PBO[22:4]: Physical buffer offset  
| Offset of the physical buffer. |
| Bits 3:0 | Reserved, must be kept at reset value. |

### 21.5.8 Graphic MMU buffer 2 configuration register (GFXMMU_B2CR)

Address offset: 0x0028  
Reset value: 0x0000 0000

| Bits 31:23 | PBBA[31:23]: Physical buffer base address  
| Base address MSB of the physical buffer. |
| Bits 22:4 | PBO[22:4]: Physical buffer offset  
| Offset of the physical buffer. |
| Bits 3:0 | Reserved, must be kept at reset value. |
21.5.9  Graphic MMU buffer 3 configuration register (GFXMMU_B3CR)

Address offset: 0x002C
Reset value: 0x0000 0000

| Bits 31:23 | PBBA[31:23]: Physical buffer base address |
| Base address MSB of the physical buffer. |
| Bits 22:4 | PBO[22:16]: Physical buffer offset |
| Offset of the physical buffer. |
| Bits 3:0 | Reserved, must be kept at reset value. |

21.5.10  Graphic MMU LUT entry x low (GFXMMU_LUTxL)

Address offset: 0x1000 + 8 * x, x = 0...1023
Reset value: 0x0000 0000

| Bits 31:24 | Reserved, must be kept at reset value. |
| Bits 23:16 | LVB[7:0]: Last Valid Block |
| Number of the last valid block of line number X. |
| Bits 15:8 | FVB[7:0]: First Valid Block |
| Number of the first valid block of line number x. |
| Bits 7:1 | Reserved, must be kept at reset value. |
| Bit 0 | EN: Enable |
| Line enable. |
| 0: Line is disabled (no MMU evaluation is performed) |
| 1: Line is enabled (MMU evaluation is performed) |

21.5.11  Graphic MMU LUT entry x high (GFXMMU_LUTxH)

Address offset: 0x1000 + 8 * x + 4, x = 0...1023
Reset value: 0x0000 0000
| Bits 31:22 | Reserved, must be kept at reset value. |
| Bits 21:4 | **LO[21:4]**: Line offset  |
| Line offset of line number x (i.e. offset of block 0 of line x) |
| Bits 3:0 | Reserved, must be kept at reset value. |
### 21.5.12 Graphic MMU register map

The following table summarizes the graphic MMU registers. Refer to the register boundary addresses table for the graphic MMU register base address.

#### Table 130. Graphic MMU register map and reset values

| Offset   | Register   | Bits          | Description               | Reset value
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>GFXMMU_CR</td>
<td>31:0</td>
<td>PBBA[31:23], PBO[22:4]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0004</td>
<td>GFXMMU_SR</td>
<td>31:0</td>
<td>PBBA[31:23], PBO[22:4]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0008</td>
<td>GFXMMU_FCR</td>
<td>31:0</td>
<td>PBBA[31:23], PBO[22:4]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x000C</td>
<td>GFXMMU_CCR</td>
<td>31:0</td>
<td>PBBA[31:23], PBO[22:4]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0010</td>
<td>GFXMMU_DVR</td>
<td>31:0</td>
<td>PBBA[31:23], PBO[22:4]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0014</td>
<td>GFXMMU_B0CR</td>
<td>31:0</td>
<td>PBBA[31:23], PBO[22:4]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0018</td>
<td>GFXMMU_B1CR</td>
<td>31:0</td>
<td>PBBA[31:23], PBO[22:4]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x001C</td>
<td>GFXMMU_B2CR</td>
<td>31:0</td>
<td>PBBA[31:23], PBO[22:4]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0020</td>
<td>GFXMMU_B3CR</td>
<td>31:0</td>
<td>PBBA[31:23], PBO[22:4]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x002C</td>
<td>GFXMMU_LUT0L</td>
<td>31:0</td>
<td>LVB[7:0], FVB[7:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0030</td>
<td>GFXMMU_LUT0H</td>
<td>31:0</td>
<td>LVB[7:0], FVB[7:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0034</td>
<td>GFXMMU_LUT1L</td>
<td>31:0</td>
<td>LVB[7:0], FVB[7:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0038</td>
<td>GFXMMU_LUT1H</td>
<td>31:0</td>
<td>LVB[7:0], FVB[7:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x003C</td>
<td>GFXMMU_LUT2L</td>
<td>31:0</td>
<td>LVB[7:0], FVB[7:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0040</td>
<td>GFXMMU_LUT2H</td>
<td>31:0</td>
<td>LVB[7:0], FVB[7:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0044</td>
<td>GFXMMU_LUT3L</td>
<td>31:0</td>
<td>LVB[7:0], FVB[7:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0048</td>
<td>GFXMMU_LUT3H</td>
<td>31:0</td>
<td>LVB[7:0], FVB[7:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
22 Cyclic redundancy check calculation unit (CRC)

22.1 Introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

22.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  \[X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^9 + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1\]
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-, 16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data
22.3 CRC functional description

22.3.1 CRC block diagram

Figure 104. CRC calculation unit block diagram

22.3.2 CRC internal signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crc_hclk</td>
<td>Digital input</td>
<td>AHB clock</td>
</tr>
</tbody>
</table>

22.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit access is allowed.

The duration of the computation depends on data width:
- 4 AHB clock cycles for 32-bit
- 2 AHB clock cycles for 16-bit
- 1 AHB clock cycles for 8-bit

An input buffer allows a second data to be immediately written without waiting for any wait states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.
The input data can be reversed, to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example: input data 0x1A2B3C4D is used for CRC calculation as:
- 0x58D43CB2 with bit-reversal done by byte
- 0xD458B23C with bit-reversal done by half-word
- 0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register. The operation is done at bit level: for example, output data 0x11223344 is converted into 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

**Polynomial programmability**

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size cannot be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.
## 22.4 CRC registers

### 22.4.1 CRC data register (CRC_DR)

Address offset: 0x00  
Reset value: 0xFFFF FFFF

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-16</td>
<td>DR[31:16]</td>
</tr>
<tr>
<td>15-0</td>
<td>DR[15:0]</td>
</tr>
</tbody>
</table>

Bits 31:0 **DR[31:0]**: Data register bits  
This register is used to write new data to the CRC calculator.  
It holds the previous CRC calculation result when it is read.  
If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

### 22.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-16</td>
<td>IDR[31:16]</td>
</tr>
<tr>
<td>15-0</td>
<td>IDR[15:0]</td>
</tr>
</tbody>
</table>

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits  
These bits can be used as a temporary storage location for four bytes.  
This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register.
22.4.3 CRC control register (CRC CR)

Address offset: 0x08
Reset value: 0x0000 0000

| Bit 31:8 | Reserved, must be kept at reset value. |
| Bit 7  | REV_OUT: Reverse output data  |
|        | This bit controls the reversal of the bit order of the output data. |
|        | 0: Bit order not affected  |
|        | 1: Bit-reversed output format  |
| Bit 6:5 | REV_IN[1:0]: Reverse input data  |
|        | These bits control the reversal of the bit order of the input data  |
|        | 00: Bit order not affected  |
|        | 01: Bit reversal done by byte  |
|        | 10: Bit reversal done by half-word  |
|        | 11: Bit reversal done by word  |

| Bit 4:3 | POLYSIZE[1:0]: Polynomial size  |
|        | These bits control the size of the polynomial.  |
|        | 00: 32 bit polynomial  |
|        | 01: 16 bit polynomial  |
|        | 10: 8 bit polynomial  |
|        | 11: 7 bit polynomial  |

| Bit 0  | RESET: RESET bit  |
|        | This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware  |

22.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10
Reset value: 0xFFFF FFFF

| Bit 31:16 | CRC_INIT[31:16] |
|           | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| Bit 15:0 | CRC_INIT[15:0] |
|           | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
22.4.5 CRC polynomial (CRC_POL)
Address offset: 0x14
Reset value: 0x04C1 1DB7

Bits 31:0 **CRC_INIT[31:0]**: Programmable initial CRC value
This register is used to write the CRC initial value.

### 22.4.6 CRC register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CRC_DR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DR[31:0]</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0x04</td>
<td>CRC_IDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IDR[31:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x06</td>
<td>CRC_CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>REV_OUT</td>
<td>REV_IN[1:0]</td>
<td>POLYSIZE[1:0]</td>
<td>RESET</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x10</td>
<td>CRC_INIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CRC_INIT[31:0]</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0x14</td>
<td>CRC_POL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>POL[31:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 132. CRC register map and reset values

Refer to *Section 2.3 on page 128* for the register boundary addresses.
23 Flexible memory controller (FMC)

The flexible memory controller (FMC) includes three memory controllers:
- The NOR/PSRAM memory controller
- The NAND memory controller
- The Synchronous DRAM (SDRAM/Mobile LPDDR SDRAM) controller

23.1 FMC main features

The FMC functional block makes the interface with: synchronous and asynchronous static memories, SDRAM memories, and NAND flash memory. Its main purposes are:
- to translate AXI transactions into the appropriate external device protocol
- to meet the access time requirements of the external memory devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique Chip Select. The FMC performs only one access at a time to an external device.

The main features of the FMC controller are the following:
- Interface with static-memory mapped devices including:
  - Static random access memory (SRAM)
  - NOR Flash memory/OneNAND Flash memory
  - PSRAM (4 memory banks)
  - NAND Flash memory with ECC hardware to check up to 8 Kbytes of data
- Interface with synchronous DRAM (SDRAM/Mobile LPDDR SDRAM) memories
- Burst mode support for faster access to synchronous devices such as NOR Flash memory, PSRAM and SDRAM)
- Programmable continuous clock output for asynchronous and synchronous accesses
- 8-,16- or 32-bit wide data bus
- Independent Chip Select control for each memory bank
- Independent configuration for each memory bank
- Write enable and byte lane select outputs for use with PSRAM, SRAM and SDRAM devices
- External asynchronous wait control
- Write FIFO with 16 x32-bit depth
  The Write FIFO is common to all memory controllers and consists of:
  - a Write Data FIFO which stores the data to be written to the memory
  - a Write Address FIFO which stores the address (up to 28 bits) plus the data size (up to 2 bits). When operating in Burst mode, only the start address is stored except when crossing a page boundary (for PSRAM and SDRAM). In this case, the burst is broken into two FIFO entries.
- Cacheable Read FIFO with 6 x64-bit depth (6 x14-bit address tag) for SDRAM controller.

At startup the FMC pins must be configured by the user application. The FMC I/O pins which are not used by the application can be used for other purposes.
The FMC registers that define the external device type and associated characteristics are set at boot time and do not change until the next reset or power-up. However, only a few bits can be changed on-the-fly:
- ECCEN and PBKEN bits in the FMC_PCR register
- IFS, IRS and ILS bits in the FMC_SR register
- MODE[2:0], CTB1/CTB2, NRFS and MRD bits in the FMC_SDCMR register
- REIE and CRE bits in the FMC_SDRTR register.

Follow the below sequence to modify parameters while the FMC is enabled:
1. First disable the FMC controller to prevent further accesses to any memory controller while the register is modified.
2. Update all required configurations.
3. Enable the FMC controller again.

When the SDRAM controller is used, if the SDCLK Clock ratio or refresh rate has to be modified after initialization phase, the following procedure must be followed.
1. Put the SDRAM device in Self-refresh mode.
2. Disable the FMC controller by resetting the FMCEN bit in the FMC_BCR1 register.
3. Update the required parameters.
4. Enable the FMC controller once all parameters have been updated.
5. Then, send the Clock Configuration Enable command to exit Self-fresh mode.

23.2 **FMC block diagram**

The FMC consists of the following main blocks:
- The NOR Flash/PSRAM/SRAM controller
- The NAND controller
- The SDRAM controller
- The AXI interface
- The AHB interface (including the FMC configuration registers)

The block diagram is shown in the figure below.
Figure 105. FMC block diagram

- NOR/PSRAM signals
  - FMC_NE[4:1]
  - FMC_NL (or NADV)
  - FMC_NBL[3:0]
  - FMC_D[31:0]
  - FMC_NOE
  - FMC_NWE
  - FMC_NCE
  - FMC_INT
  - FMC_NWAIT

- NOR / SRAM shared signals
  - FMC_A[25:0]
  - FMC_D[31:0]

- Shared signals
  - FMC_CLK
  - FMC_SDNWE
  - FMC_SDCKE[1:0]
  - FMC_SDNE[1:0]

- NAND signals
  - FMC_NRAS
  - FMC_NCAS

- SDRAM signals
  - FMC_SDCLK
  - FMC_SDNWE
  - FMC_SDCKE[1:0]
  - FMC_SDNE[1:0]
  - FMC_NRAS
  - FMC_NCAS
23.3 FMC internal signals

Table 133 gives the list of FMC internal signals. FMC pins (or external signals) are described in Section 23.7.1: External memory interface signals.

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fmc_it</td>
<td>Digital output</td>
<td>FMC interrupt</td>
</tr>
<tr>
<td>fmc_ker_ck</td>
<td>Digital input</td>
<td>FMC kernel clock</td>
</tr>
<tr>
<td>fmc_hclk</td>
<td>Digital input</td>
<td>FMC interface clock</td>
</tr>
</tbody>
</table>

23.4 AHB interface

The AHB slave interface allows internal CPUs to configure the FMC registers.

The AHB clock (fmc_hclk) is the reference clock for the FMC register accesses.

23.5 AXI interface

The AXI slave interface allows internal CPUs and other bus master peripherals to access the external memories.

AXI transactions are translated into the external device protocol. As the AXI data bus is 64-bit wide, the AXI transactions might be split into several consecutive 32-, 16- or 8-bit accesses according to data size accesses. The FMC Chip Select (FMC_NEx) does not toggle between consecutive accesses except in case of accesses in mode D when the Extended mode is enabled.

The FMC generates an AXI slave error when one of the following conditions is met:

- Reading or writing to an FMC bank (Bank 1 to 4) which is not enabled.
- Reading or writing to the NOR Flash bank while the FACCEN bit is reset in the FMC_BCRx register.
- Writing to a write protected SDRAM bank (WP bit set in the FMC_SDCRx register).
- Violation of the SDRAM address range (access to reserved address range)
- Attempting to read/write access from/to SDRAM bank when it is not yet initialized

The FMC generates an AXI decoder error when ADDR[31:28] address bits are not supported by the FMC bank base address following the BMAP[1:0] bits configuration.

The kernel clock for the FMC controller is the asynchronous fmc_ker_ck clock (refer to Section 8: Reset and clock control (RCC) for fmc_ker_ck clock source selection).
23.5.1 Supported memories and transactions

General transaction rules

The requested AXI transaction data size can be 8-, 16-, 32- or 64-bit wide whereas the accessed external device has a fixed data width. The AXI transaction size depends on memory region MPU attributes. This may lead to inconsistent transfers.

Therefore, some simple transaction rules can be followed, depending on AXI transaction size versus memory data size:

- AXI transaction size and memory data size are equal to prevent issues from occurring.
- AXI transaction size is greater than the memory size:
  In this case, the FMC splits the AXI transaction into smaller consecutive memory accesses to meet the external data width.
  The memory region MPU attribute must be configured as “Device” or “strongly-ordered” in order to prevent issues on devices that do not have the byte select feature. As an example, if byte accesses are performed to 16-bit memories with the MPU attribute set to “Normal memory”, the AXI transaction size is 64 bits and the FMC splits the transaction into four consecutive accesses while only one access is expected instead of four.
- AXI transaction size is smaller than the memory size:
  The transfer may or not be consistent depending on the type of external device:
  - Accesses to devices that have the byte select feature (SRAM, ROM, PSRAM, SDRAM)
    In this case, the FMC allows read/write transactions and accesses the right data through its NBL[3:0] byte lanes.
    Bytes to be written are addressed by NBL[3:0].
    All memory bytes are read (NBL[3:0] are driven low during read transaction) and the useless ones are discarded.
  - Accesses to devices that do not have the byte select feature (NOR and NAND Flash memories)
    This situation occurs when a byte access is requested to a 16-bit wide Flash memory. Since the device cannot be accessed in Byte mode (only 16-bit words can be read/written from/to the Flash memory), write transactions are not allowed while read transactions are allowed (the controller reads the entire 16-bit memory word and uses only the required byte).

Wrap support for NOR Flash/PSRAM and SDRAM

The synchronous memories must be configured in Linear burst mode of undefined length as not all masters can issue wrap transactions.

If a master generates a wrap transaction:
- The read is split into two linear burst transactions.
- The write is split into two linear burst transactions if the write FIFO is enabled and into several linear burst transactions if the write FIFO is disabled.

Configuration registers

The FMC can be configured through a set of registers. Refer to Section 23.7.6, for a detailed description of the NOR Flash/PSRAM controller registers. Refer to Section 23.8.7.
for a detailed description of the NAND Flash registers and to Section 23.9.5 for a detailed description of the SDRAM controller registers.

23.6 External device address mapping

From the FMC point of view, the external memory is divided into fixed-size banks of 256 Mbytes each (see Figure 106):

- Bank 1 is used to address up to 4 NOR Flash memory or PSRAM devices. This bank is split into 4 NOR/PSRAM subbanks with 4 dedicated Chip Selects, as follows:
  - Bank 1 - NOR/PSRAM 1
  - Bank 1 - NOR/PSRAM 2
  - Bank 1 - NOR/PSRAM 3
  - Bank 1 - NOR/PSRAM 4
- Bank 2 and Bank 4 are not used by the FMC.
- Bank 3 is used to address NAND Flash memory devices. The MPU memory attribute for this space must be reconfigured by software to Device.
- Bank 5 and 6 are used to address SDRAM devices (1 device per bank).

For each bank the type of memory to be used can be configured by the user application through the Configuration register.
The FMC bank mapping can be modified through the BMAP[1:0] bits in the FMC_BCR1 register. Table 134 shows the configuration to swap the NOR/PSRAM bank with SDRAM banks.

### Table 134. FMC bank mapping options

<table>
<thead>
<tr>
<th>Start - End address</th>
<th>BMAP[1:0]=00 (Default mapping)</th>
<th>BMAP[1:0]=01 NOR/PSRAM and SDRAM banks swapped</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x6000 0000 - 0xFFFF FFFF</td>
<td>NOR/PSRAM bank</td>
<td>SDRAM bank1</td>
</tr>
<tr>
<td>0x7000 0000 - 0x7FFF FFFF</td>
<td>Not used by FMC</td>
<td></td>
</tr>
<tr>
<td>0x8000 0000 - 0x8FFF FFFF</td>
<td>NAND bank</td>
<td>NAND bank</td>
</tr>
<tr>
<td>0x9000 0000 - 0x9FFF FFFF</td>
<td>Not used by FMC</td>
<td></td>
</tr>
<tr>
<td>0xC000 0000 - 0xCFFF FFFF</td>
<td>SDRAM bank1</td>
<td>NOR/PSRAM bank</td>
</tr>
<tr>
<td>0xD000 0000 - 0xDFFF FFFF</td>
<td>SDRAM bank2</td>
<td>SDRAM bank2</td>
</tr>
</tbody>
</table>

23.6.1 **NOR/PSRAM address mapping**

ADDR[27:26] bits are used to select one of the four memory banks as shown in Table 135.
The ADDR[25:0] bits contain the external memory address. Since ADDR is a byte address whereas the memory is addressed at word level, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

### Table 135. NOR/PSRAM bank selection

<table>
<thead>
<tr>
<th>ADDR<a href="1">27:26</a></th>
<th>Selected bank</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Bank 1 - NOR/PSRAM 1</td>
</tr>
<tr>
<td>01</td>
<td>Bank 1 - NOR/PSRAM 2</td>
</tr>
<tr>
<td>10</td>
<td>Bank 1 - NOR/PSRAM 3</td>
</tr>
<tr>
<td>11</td>
<td>Bank 1 - NOR/PSRAM 4</td>
</tr>
</tbody>
</table>

1. ADDR are internal address lines that are translated to external memory.

### Table 136. NOR/PSRAM External memory address

<table>
<thead>
<tr>
<th>Memory width(1)</th>
<th>Data address issued to the memory</th>
<th>Maximum memory capacity (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit</td>
<td>ADDR[25:0]</td>
<td>64 Mbytes x 8 = 512 Mbit</td>
</tr>
<tr>
<td>16-bit</td>
<td>ADDR[25:1] &gt;&gt; 1</td>
<td>64 Mbytes/2 x 16 = 512 Mbit</td>
</tr>
<tr>
<td>32-bit</td>
<td>ADDR[25:2] &gt;&gt; 2</td>
<td>64 Mbytes/4 x 32 = 512 Mbit</td>
</tr>
</tbody>
</table>

1. In case of a 16-bit external memory width, the FMC will internally use ADDR[25:1] to generate the address for external memory FMC_A[24:0]. In case of a 32-bit memory width, the FMC will internally use ADDR[25:2] to generate the external address. Whatever the external memory width, FMC_A[0] should be connected to external memory address A[0].

#### 23.6.2 NAND Flash memory address mapping

The NAND bank is divided into memory areas as indicated in Table 137.

### Table 137. NAND memory mapping and timing registers

<table>
<thead>
<tr>
<th>Start address</th>
<th>End address</th>
<th>FMC bank</th>
<th>Memory space</th>
<th>Timing register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x8800 0000</td>
<td>0x8BFF FFFF</td>
<td>Bank 3 - NAND Flash</td>
<td>Attribute</td>
<td>FMC_PATT (0x8C)</td>
</tr>
<tr>
<td>0x8000 0000</td>
<td>0x83FF FFFF</td>
<td>Common</td>
<td>FMC_PMEM (0x88)</td>
<td></td>
</tr>
</tbody>
</table>

For NAND Flash memory, the common and attribute memory spaces are subdivided into three sections (see in Table 138 below) located in the lower 256 Kbytes:
- Data section (first 64 Kbytes in the common(attribute memory space)
- Command section (second 64 Kbytes in the common / attribute memory space)
- Address section (next 128 Kbytes in the common / attribute memory space)

### Table 138. NAND bank selection

<table>
<thead>
<tr>
<th>Section name</th>
<th>ADDR[17:16]</th>
<th>Address range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address section</td>
<td>1X</td>
<td>0x020000-0x03FFFF</td>
</tr>
<tr>
<td>Command section</td>
<td>01</td>
<td>0x010000-0x01FFFF</td>
</tr>
<tr>
<td>Data section</td>
<td>00</td>
<td>0x000000-0x0FFFF</td>
</tr>
</tbody>
</table>
The application software uses the 3 sections to access the NAND Flash memory:

- **To send a command to NAND Flash memory**, the software must write the command value to any memory location in the command section.
- **To specify the NAND Flash address that must be read or written**, the software must write the address value to any memory location in the address section. Since an address can be 4 or 5 bytes long (depending on the actual memory size), several consecutive write operations to the address section are required to specify the full address.
- **To read or write data**, the software reads or writes the data from/to any memory location in the data section.

Since the NAND Flash memory automatically increments addresses, there is no need to increment the address of the data section to access consecutive memory locations.

### 23.6.3 SDRAM address mapping

Two SDRAM banks are available as indicated in Table 139.

<table>
<thead>
<tr>
<th>Selected bank</th>
<th>Control register</th>
<th>Timing register</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDRAM Bank1</td>
<td>FMC_SDCR1</td>
<td>FMC_SDTR1</td>
</tr>
<tr>
<td>SDRAM Bank2</td>
<td>FMC_SDCR2</td>
<td>FMC_SDTR2</td>
</tr>
</tbody>
</table>

Table 140 shows SDRAM mapping for a 13-bit row and an 11-bit column configuration.

<table>
<thead>
<tr>
<th>Memory width(1)</th>
<th>Internal bank</th>
<th>Row address</th>
<th>Column address(2)</th>
<th>Maximum memory capacity (Mbytes)</th>
</tr>
</thead>
</table>

1. When interfacing with a 16-bit memory, the FMC internally uses the ADDR[11:1] internal address lines to generate the external address. When interfacing with a 32-bit memory, the FMC internally uses ADDR[12:2] lines to generate the external address. Whatever the memory width, FMC_A[0] has to be connected to the external memory address A[0].

2. The AutoPrecharge is not supported. FMC_A[10] must be connected to the external memory address A[10] but it will be always driven low.

The ADDR[27:0] bits are translated into an external SDRAM address depending on the SDRAM controller configuration:

- Data size: 8, 16 or 32 bits
- Row size: 11, 12 or 13 bits
- Column size: 8, 9, 10 or 11 bits
- Number of internal banks: two or four internal banks
The following tables show the SDRAM address mapping versus the SDRAM controller configuration.

### Table 141. SDRAM address mapping with 8-bit data bus width\(^{(1)}\)(\(^{(2)}\))

<table>
<thead>
<tr>
<th>Row size configuration</th>
<th>ADDR (Internal Address Lines)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to '0'.
2. Access to Reserved (Res.) address range generates an AXI slave error.
### Table 142. SDRAM address mapping with 16-bit data bus width\(^{(1)}\)(\(^{(2)}\))

| Row size Configuration | ADDR(address Lines) | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|------------------------|---------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 11-bit row size        | Res.                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| configuration          | Bank [1:0]          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|                        | Row[10:0]           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|                        | Column[7:0]         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|                        | BM0 (3)             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 12-bit row size        | Res.                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| configuration          | Bank [1:0]          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|                        | Row[11:0]           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|                        | Column[7:0]         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|                        | BM0                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 13-bit row size        | Res.                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| configuration          | Bank [1:0]          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|                        | Row[12:0]           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|                        | Column[7:0]         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|                        | BM0                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to '0'.
2. Access to Reserved space (Res.) generates an AXI Slave error.
3. BM0: is the byte mask for 16-bit access.
23.7 NOR Flash/PSRAM controller

The FMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM and ROM
  - 8 bits
  - 16 bits
  - 32 bits
- PSRAM (Cellular RAM)
  - Asynchronous mode
  - Burst mode for synchronous accesses with configurable option to split burst access when crossing boundary page for CRAM 1.5.
  - Multiplexed or non-multiplexed
- NOR Flash memory
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed
The FMC outputs a unique Chip Select signal, NE[4:1], per bank. All the other signals (addresses, data and control) are shared.

The FMC supports a wide range of devices through a programmable timings among which:

- Programmable wait states (up to 15)
- Programmable bus turnaround cycles (up to 15)
- Programmable output enable and write enable delays (up to 15)
- Independent read and write timings and protocol to support the widest variety of memories and timings
- Programmable continuous clock (FMC_CLK) output.

The FMC output Clock (FMC_CLK) is a sub-multiple of the fmc_ker_ck clock. It can be delivered to the selected external device either during synchronous accesses only or during asynchronous and synchronous accesses depending on the CCKEN bit configuration in the FMC_BCR1 register:

- If the CCLKEN bit is reset, the FMC generates the clock (FMC_CLK) only during synchronous accesses (Read/write transactions).
- If the CCLKEN bit is set, the FMC generates a continuous clock during asynchronous and synchronous accesses. To generate the FMC_CLK continuous clock, Bank 1 must be configured in Synchronous mode (see Section 23.7.6: NOR/PSRAM controller registers). Since the same clock is used for all synchronous memories, when a continuous output clock is generated and synchronous accesses are performed, the AXI data size has to be the same as the memory data width (MWID) otherwise the FMC_CLK frequency will be changed depending on AXI data transaction (refer to Section 23.7.5: Synchronous transactions for FMC_CLK divider ratio formula).

The size of each bank is fixed and equal to 64 Mbytes. Each bank is configured through dedicated registers (see Section 23.7.6: NOR/PSRAM controller registers).

The programmable memory parameters include access times (see Table 144) and support for wait management (for PSRAM and NOR Flash memory accessed in Burst mode).

Table 144. Programmable NOR/PSRAM access parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Access mode</th>
<th>Unit</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address setup</td>
<td>Duration of the address setup phase</td>
<td>Asynchronous</td>
<td>FMC clock cycle (fmc_ker_ck)</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>Address hold</td>
<td>Duration of the address hold phase</td>
<td>Asynchronous, muxed I/Os</td>
<td>FMC clock cycle (fmc_ker_ck)</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>Data setup</td>
<td>Duration of the data setup phase</td>
<td>Asynchronous</td>
<td>FMC clock cycle (fmc_ker_ck)</td>
<td>1</td>
<td>256</td>
</tr>
<tr>
<td>Bust turn</td>
<td>Duration of the bus turnaround phase</td>
<td>Asynchronous and synchronous read</td>
<td>FMC clock cycle (fmc_ker_ck)</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>Clock divide ratio</td>
<td>Number of FMC clock cycles (fmc_ker_ck) to build one memory clock cycle (CLK)</td>
<td>Synchronous</td>
<td>FMC clock cycle (fmc_ker_ck)</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>Data latency</td>
<td>Number of clock cycles to issue to the memory before the first data of the burst</td>
<td>Synchronous</td>
<td>Memory clock cycle (fmc_ker_ck)</td>
<td>2</td>
<td>17</td>
</tr>
</tbody>
</table>
23.7.1 External memory interface signals

*Table 145, Table 146 and Table 147 list the signals that are typically used to interface with NOR Flash memory, SRAM and PSRAM.*

*Note: The prefix “N” identifies the signals which are active low.*

**NOR Flash memory, non-multiplexed I/Os**

<table>
<thead>
<tr>
<th>FMC pin name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>O</td>
<td>Clock (for synchronous access)</td>
</tr>
<tr>
<td>A[25:0]</td>
<td>O</td>
<td>Address bus</td>
</tr>
<tr>
<td>D[31:0]</td>
<td>I/O</td>
<td>Bidirectional data bus</td>
</tr>
<tr>
<td>NE[x]</td>
<td>O</td>
<td>Chip Select, x = 1..4</td>
</tr>
<tr>
<td>NOE</td>
<td>O</td>
<td>Output enable</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NL(=NADV)</td>
<td>O</td>
<td>Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)</td>
</tr>
<tr>
<td>NWAIT</td>
<td>I</td>
<td>NOR Flash wait input signal to the FMC</td>
</tr>
</tbody>
</table>

The maximum capacity is 512 Mbits (26 address lines).

**NOR Flash memory, 16-bit multiplexed I/Os**

<table>
<thead>
<tr>
<th>FMC pin name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>O</td>
<td>Clock (for synchronous access)</td>
</tr>
<tr>
<td>AD[15:0]</td>
<td>I/O</td>
<td>16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)</td>
</tr>
<tr>
<td>NE[x]</td>
<td>O</td>
<td>Chip Select, x = 1..4</td>
</tr>
<tr>
<td>NOE</td>
<td>O</td>
<td>Output enable</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NL(=NADV)</td>
<td>O</td>
<td>Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)</td>
</tr>
<tr>
<td>NWAIT</td>
<td>I</td>
<td>NOR Flash wait input signal to the FMC</td>
</tr>
</tbody>
</table>

The maximum capacity is 512 Mbits.
PSRAM/SRAM, non-multiplexed I/Os

Table 147. Non-multiplexed I/Os PSRAM/SRAM

<table>
<thead>
<tr>
<th>FMC pin name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>O</td>
<td>Clock (only for PSRAM synchronous access)</td>
</tr>
<tr>
<td>A[25:0]</td>
<td>O</td>
<td>Address bus</td>
</tr>
<tr>
<td>D[31:0]</td>
<td>I/O</td>
<td>Data bidirectional bus</td>
</tr>
<tr>
<td>NE[x]</td>
<td>O</td>
<td>Chip Select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))</td>
</tr>
<tr>
<td>NOE</td>
<td>O</td>
<td>Output enable</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NL(= NADV)</td>
<td>O</td>
<td>Address valid only for PSRAM input (memory signal name: NADV)</td>
</tr>
<tr>
<td>NWAIT</td>
<td>I</td>
<td>PSRAM wait input signal to the FMC</td>
</tr>
<tr>
<td>NBL[3:0]</td>
<td>O</td>
<td>Byte lane output. Byte 0 to Byte 3 control (Upper and lower byte enable)</td>
</tr>
</tbody>
</table>

The maximum capacity is 512 Mbits.

PSRAM, 16-bit multiplexed I/Os

Table 148. 16-Bit multiplexed I/O PSRAM

<table>
<thead>
<tr>
<th>FMC pin name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>O</td>
<td>Clock (for synchronous access)</td>
</tr>
<tr>
<td>AD[15:0]</td>
<td>I/O</td>
<td>16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)</td>
</tr>
<tr>
<td>NE[x]</td>
<td>O</td>
<td>Chip Select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))</td>
</tr>
<tr>
<td>NOE</td>
<td>O</td>
<td>Output enable</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NL(= NADV)</td>
<td>O</td>
<td>Address valid PSRAM input (memory signal name: NADV)</td>
</tr>
<tr>
<td>NWAIT</td>
<td>I</td>
<td>PSRAM wait input signal to the FMC</td>
</tr>
<tr>
<td>NBL[1:0]</td>
<td>O</td>
<td>Byte lane output. Byte 0 and Byte 1 control (upper and lower byte enable)</td>
</tr>
</tbody>
</table>

The maximum capacity is 512 Mbits (26 address lines).
### Supported memories and transactions

Table 149 below shows an example of the supported devices, access modes and transactions when the memory data bus is 16-bit wide for NOR Flash memory, PSRAM and SRAM. The transactions not allowed (or not supported) by the FMC are shown in gray in this example.

<table>
<thead>
<tr>
<th>Device</th>
<th>Mode</th>
<th>R/W</th>
<th>AXI data size</th>
<th>Memory data size</th>
<th>Allowed/not allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NOR Flash</strong></td>
<td>Asynchronous</td>
<td>R</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>8</td>
<td>16</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>64</td>
<td>16</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>64</td>
<td>16</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>-</td>
<td>16</td>
<td>N</td>
<td>Mode is not supported</td>
</tr>
<tr>
<td><strong>PSRAM</strong></td>
<td>Asynchronous</td>
<td>R</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td>Use of byte lanes NBL[1:0]</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>64</td>
<td>16</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>64</td>
<td>16</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>-</td>
<td>16</td>
<td>N</td>
<td>Mode is not supported</td>
</tr>
<tr>
<td><strong>SRAM</strong></td>
<td>Synchronous</td>
<td>R</td>
<td>8</td>
<td>16</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>R</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>R</td>
<td>32/64</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>W</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td>Use of byte lanes NBL[1:0]</td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>W</td>
<td>16/32/64</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>
23.7.3 General timing rules

Signal synchronization is performed as follows:

- All controller output signals change on the rising edge of the fmc_ker_ck clock.
- In Synchronous read and write modes, all output signals change on the rising edge of the fmc_ker_ck clock. Whatever the CLKDIV value, all outputs change as follows:
  - NOEL/NWEL/ NEL/NADV/ NADVH /NBL/ Address valid outputs change on the falling edge of FMC_CLK clock.
  - NOEH/ NWEH / NEH/ NOEH/NBLH/ Address invalid outputs change on the rising edge of FMC_CLK clock.

23.7.4 NOR Flash/PSRAM controller asynchronous transactions

Asynchronous transactions on static memories (NOR Flash memory, PSRAM, SRAM) are performed as follows:

- Signals are synchronized by the internal clock. This clock is not issued to the memory.
- The FMC always samples the data before deasserting the Chip Select signal. This guarantees that the memory data hold timing constraint is met (minimum Chip Enable high to data transition is usually 0 ns)
- If the Extended mode is enabled (EXTMOD bit is set in the FMC_BCRx register), up to four extended modes (A, B, C and D) are available. It is possible to mix A, B, C and D modes for read and write operations. For example, read operation can be performed in mode A and write in mode B.
- If the Extended mode is disabled (EXTMOD bit is reset in the FMC_BCRx register), the FMC can operate in Mode1 or Mode2 as follows:
  - Mode 1 is the default mode when SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01 in the FMC_BCRx register)
  - Mode 2 is the default mode when NOR memory type is selected (MTYP = 0x10 in the FMC_BCRx register).

---

Table 149. NOR Flash/PSRAM: Example of supported memories and transactions(1) (continued)

<table>
<thead>
<tr>
<th>Device</th>
<th>Mode</th>
<th>R/W</th>
<th>AXI data size</th>
<th>Memory data size</th>
<th>Allowed/not allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM and ROM</td>
<td>Asynchronous</td>
<td>R</td>
<td>8/16</td>
<td>16</td>
<td>Y</td>
<td>Use of byte lanes NBL[1:0]</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>8/16</td>
<td>16</td>
<td>Y</td>
<td>Use of byte lanes NBL[1:0]</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>64</td>
<td>16</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>64</td>
<td>16</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
</tbody>
</table>

1. NBL[1:0] are also driven by AXI write strobes.
Mode 1 - SRAM/PSRAM (CRAM)

The next figures show the read and write transactions for the supported modes followed by the required configuration of FMC_BCRx, and FMC_BTRx/FMC_BWTRx registers.

**Figure 107. Mode 1 read access waveforms**
The fmc_ker_ck cycle at the end of the write transaction helps guarantee the address and data hold time after the NWE rising edge. Due to the presence of this fmc_ker_ck cycle, the DATAST value must be greater than zero (DATAST > 0).

**Table 150. FMC_BCRx bitfields (mode 1)**

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>Bit number</td>
<td>Bit name</td>
<td>Value to set</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>Don’t care</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>As needed, exclude 0x2 (NOR Flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXE</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

Table 150. FMC_BCRx bitfields (mode 1) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST+1 fmc_ker_ck cycles for write accesses, DATAST fmc_ker_ck cycles for read accesses).</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles). Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>
Mode A - SRAM/PSRAM (CRAM) OE toggling

Figure 109. Mode A read access waveforms

1. NBL[3:0] are driven low during the read access
The differences compared with Mode 1 are the toggling of NOE and the independent read and write timings.

Table 152. FMC_BCRx bitfields (mode A)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x1</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
</tbody>
</table>
### Table 152. FMC_BCRx bitfields (mode A) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCE0</td>
<td>Don’t care</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>As needed, exclude 0x2 (NOR Flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 153. FMC_BTRx bitfields (mode A)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET = 0.</td>
</tr>
</tbody>
</table>

### Table 154. FMC_BWTRx bitfields (mode A)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET = 0.</td>
</tr>
</tbody>
</table>
Mode 2/B - NOR Flash

Figure 111. Mode 2 and mode B read access waveforms

Figure 112. Mode 2 write access waveforms
The differences with Mode 1 are the toggling of NWE and the independent read and write timings when extended mode is set (mode B).

Table 155. FMC_BCRx bitfields (mode 2/B)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x1 for mode B, 0x0 for mode 2</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don't care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
</tbody>
</table>
### Table 155. FMC_BCRx bitfields (mode 2/B) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>0x2 (NOR Flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 156. FMC_BTRx bitfields (mode 2/B)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x1 if Extended mode is set</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don't care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don't care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the access second phase (DATAST fmc_ker_ck cycles) for read accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don't care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the access first phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>

### Table 157. FMC_BWTRx bitfields (mode 2/B)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x1 if Extended mode is set</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don't care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don't care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the access second phase (DATAST fmc_ker_ck cycles) for write accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don't care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the access first phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>
**Note:** The FMC_BWTRx register is valid only if the Extended mode is set (mode B), otherwise its content is don’t care.

**Mode C - NOR Flash - OE toggling**

**Figure 114. Mode C read access waveforms**

**Figure 115. Mode C write access waveforms**
The differences compared with Mode1 are the toggling of NOE and the independent read and write timings.

Table 158. FMC_BCRx bitfields (mode C)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x1</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>0x02 (NOR Flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>
### Table 159. FMC_BTRx bitfields (mode C)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x2</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>0x0</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>0x0</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>

### Table 160. FMC_BWTRx bitfields (mode C)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x2</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>
Mode D - asynchronous access with extended address

Figure 116. Mode D read access waveforms

Figure 117. Mode D write access waveforms
The differences with Mode1 are the toggling of NOE that goes on toggling after NADV changes and the independent read and write timings.

### Table 161. FMC_BCRx bitfields (mode D)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x1</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>Set according to memory support</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>As needed</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 162. FMC_BTRx bitfields (mode D)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x3</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.</td>
</tr>
</tbody>
</table>
### Table 162. FMC_BTRx bitfields (mode D) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Duration of the middle phase of the read access (ADDHLD fmc_ker_ck cycles)</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 1.</td>
</tr>
</tbody>
</table>

### Table 163. FMC_BWTRx bitfields (mode D)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x3</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don't care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don't care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST + 1 fmc_ker_ck cycles) for write accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Duration of the middle phase of the write access (ADDHLD fmc_ker_ck cycles)</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 1.</td>
</tr>
</tbody>
</table>
Muxed mode - multiplexed asynchronous access to NOR Flash memory

The difference with Mode D is the drive of the lower address byte(s) on the data bus.
### Table 164. FMC_BCRx bitfields (Muxed mode)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x00</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x00</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>0x2 (NOR Flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x1</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 165. FMC_BTRx bitfields (Muxed mode)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST fmc_ker_ck cycles for read accesses and DATAST+1 fmc_ker_ck cycles for write accesses).</td>
</tr>
</tbody>
</table>
WAIT management in asynchronous accesses

If the asynchronous memory asserts the WAIT signal to indicate that it is not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase), programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.

The data setup phase must be programmed so that WAIT can be detected \(4 \times fmc\_ker\_ck\) cycles before the end of the memory transaction. The following cases must be considered:

1. The memory asserts the WAIT signal aligned to NOE/NWE which toggles:
   \[
   \text{DATAST} \geq (4 \times fmc\_ker\_ck) + \text{max\_wait\_assertion\_time}
   \]

2. The memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):
   \[
   \text{if } \text{max\_wait\_assertion\_time} > \text{address\_phase} + \text{hold\_phase} \text{ then:}
   \]
   \[
   \text{DATAST} \geq (4 \times fmc\_ker\_ck) + (\text{max\_wait\_assertion\_time} - \text{address\_phase} - \text{hold\_phase})
   \]
   otherwise
   \[
   \text{DATAST} \geq (4 \times fmc\_ker\_ck)
   \]
   where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

Figure 120 and Figure 121 show the number of fmc\_ker\_ck clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Duration of the middle phase of the access (ADDHLD fmc_ker_ck cycles).</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles). Minimum value for ADDSET is 1.</td>
</tr>
</tbody>
</table>
1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

**Figure 120. Asynchronous wait during a read access waveforms**

**Figure 121. Asynchronous wait during a write access waveforms**

1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.
23.7.5 Synchronous transactions

The memory clock, FMC_CLK, is a sub-multiple of fmc_ker_ck. It depends on the value of CLKDIV and the MWID/AXI data size, following the formula given below:

\[ \text{FMC_CLK divider ratio} = \max(\text{CLKDIV} + 1, \text{MWID} \times \text{AXI data size}) \]

If MWID is 16 or 8-bit, the FMC_CLK divider ratio is always defined by the programmed CLKDIV value.

If MWID is 32-bit, the FMC_CLK divider ratio depends also on AXI data size.

Example:
- If CLKDIV=1, MWID = 32 bits, AXI data size=8 bits, FMC_CLK=fmc_ker_ck/4.
- If CLKDIV=1, MWID = 16 bits, AXI data size=8 bits, FMC_CLK=fmc_ker_ck/2.

NOR Flash memories specify a minimum time from NADV assertion to FMC_CLK high. To meet this constraint, the FMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs in the middle of the NADV low pulse.

For some PSRAM memories which must be configured to Synchronous mode, during the BCR register writing, the memory attribute space must be configured to device or strongly-ordered. Once PSRAM BCR register is configured, the memory attribute of PSRAM address space can be programmed to cacheable.

Data latency versus NOR memory latency

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR Flash configuration register. The FMC does not include the clock cycle when NADV is low in the data latency count.

Caution: Some NOR Flash memories include the NADV Low cycle in the data latency count, so that the exact relation between the NOR Flash latency and the FMC DATLAT parameter can be either:
- NOR Flash latency = (DATLAT + 2) FMC_CLK clock cycles
- or NOR Flash latency = (DATLAT + 3) FMC_CLK clock cycles

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FMC samples the data and waits long enough to evaluate if the data are valid. Thus the FMC detects when the memory exits latency and real data are processed.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.
Single-burst transfer

When the selected bank is configured in Burst mode for synchronous accesses, if for example a single-burst transaction is requested on 16-bit memories, the FMC performs a burst transaction of length 1 (if the AXI transfer is 16 bits), or length 2 (if the AXI transfer is 32 bits) and deassert the Chip Select signal when the last data is strobed.

Such transfers are not the most efficient in terms of cycles compared to asynchronous read operations. Nevertheless, a random asynchronous access would first require to re-program the memory access mode, which would altogether last longer.

Cross boundary page for Cellular RAM 1.5

Cellular RAM 1.5 does not allow burst access to cross the page boundary. The FMC controller allows to split automatically the burst access when the memory page size is reached by configuring the CPSIZE bits in the FMC_BCR1 register following the memory page size.

Wait management

For synchronous NOR Flash memories, NWAIT is evaluated after the programmed latency period, which corresponds to (DATLAT+2) FMC_CLK clock cycles.

If NWAIT is active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait states are inserted until NWAIT is inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the Chip Select and output enable signals valid. It does not consider the data as valid.

In Burst mode, there are two timing configurations for the NOR Flash NWAIT signal:
- The Flash memory asserts the NWAIT signal one data cycle before the wait state (default after reset).
- The Flash memory asserts the NWAIT signal during the wait state.

The FMC supports both NOR Flash wait state configurations, for each Chip Select, thanks to the WAITCFG bit in the FMC_BCRx registers (x = 0..3).
Figure 122. Wait configuration waveforms

Figure 123. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)

1. Byte lane outputs (NBL are not shown; for NOR access, they are held high, and, for PSRAM (CRAM) access, they are held low.
### Table 166. FMC_BCRx bitfields (Synchronous multiplexed read mode)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>MC</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>No effect on synchronous read</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>As needed. (0x1 when using CRAM 1.5)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>0x0</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>to be set to 1 if the memory supports this feature, to be kept at 0 otherwise</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>no effect on synchronous read</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>to be set according to memory</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>to be set according to memory</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x1</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>Set according to memory support (NOR Flash memory)</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>0x1 or 0x2</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>As needed</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 167. FMC_BTRx bitfields (Synchronous multiplexed read mode)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Data latency</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Data latency</td>
</tr>
</tbody>
</table>
| 23:20      | CLKDIV     | 0x0 to get CLK = fmc_ker_ck  
|            |            | 0x1 to get CLK = 2 × fmc_ker_ck  
|            |            | ..  |
| 19:16      | BUSTURN    | Time between NEx high to NEx low (BUSTURN fmc_ker_ck) |
| 15:8       | DATAST     | Don’t care   |
1. The memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.
2. Byte Lane (NBL) outputs are not shown, they are held low while NEx is active.

### Table 167. FMC_BCRx bitfields (Synchronous multiplexed read mode) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Don’t care</td>
</tr>
</tbody>
</table>

### Table 168. FMC_BCRx bitfields (Synchronous multiplexed write mode)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>No effect on synchronous read</td>
</tr>
</tbody>
</table>
Table 168. FMC_BCRx bitfields (Synchronous multiplexed write mode) (continued)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>As needed. (0x1 when using CRAM 1.5)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>0x0</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>to be set to 1 if the memory supports this feature, to be kept at 0 otherwise.</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>0x1</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>0x0</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>to be set according to memory</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>no effect on synchronous write</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>Set according to memory support</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>0x1</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>As needed</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

Table 169. FMC_BTRx bitfields (Synchronous multiplexed write mode)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Data latency</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>0x0 to get CLK = fmc_ker_ck</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x1 to get CLK = 2 × fmc_ker_ck</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Don’t care</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Don’t care</td>
</tr>
</tbody>
</table>
23.7.6  NOR/PSRAM controller registers

SRAM/NOR-Flash chip-select control registers for bank x (FMC_BCRx)

Address offset: 8 * (x - 1), (x = 1 to 4)
Reset value: Block 1: 0x0000 30DB
Reset value: Block 2: 0x0000 30D2
Reset value: Block 3: 0x0000 30D2
Reset value: Block 4: 0x0000 30D2

This register contains the control information of each memory bank, used for SRAMs, PSRAM and NOR Flash memories.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30-26</th>
<th>Bit 25-24</th>
<th>Bit 23-22</th>
<th>Bit 21</th>
<th>Bit 20-16</th>
<th>Bit 15-11</th>
<th>Bit 10-8</th>
<th>Bit 7-5</th>
<th>Bit 4-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMCEN</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>ExtMod</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>WREN</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>WACFG</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>WArOL</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>BURSTEN</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>FACCEN</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>MWD</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>MTYP</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>MUXEN</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>MBKEN</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
</tbody>
</table>

Bit 31  **FMCEN**: FMC controller Enable
This bit enables/disables the FMC controller.
0: Disable the FMC controller
1: Enable the FMC controller

*Note: The FMCEN bit of the FMC_BCR2..4 registers is don’t care. It is only enabled through the FMC_BCR1 register.*

Bits 30:26  Reserved, must be kept at reset value.

Bits 25:24  **BMAP[1:0]**: FMC bank mapping
These bits allow different remap or swap of the FMC NOR/PSRAM and SDRAM banks (refer to Table 134).
00: Default mapping (refer to Figure 106 and Table 134).
01: NOR/PSRAM bank and SDRAM bank 1 are swapped.
10: Reserved
11: Reserved.

*Note: The BMAP bits of the FMC_BCR2..4 registers are don’t care. It is only enabled through the FMC_BCR1 register.*

Bits 23:22  Reserved, must be kept at reset value.

Bit 21  **WFDIS**: Write FIFO Disable
This bit disables the Write FIFO used by the FMC controller.
0: Write FIFO enabled (Default after reset)
1: Write FIFO disabled

*Note: The WFDIS bit of the FMC_BCR2..4 registers is don’t care. It is only enabled through the FMC_BCR1 register.*
Bit 20 CCLKEN: Continuous Clock Enable
This bit enables the FMC_CLK clock output to external memory devices.
0: The FMC_CLK is only generated during the synchronous memory access (read/write transaction). The FMC_CLK clock ratio is specified by the programmed CLKDIV value in the FMC_BCRx register (default after reset).
1: The FMC_CLK is generated continuously during asynchronous and synchronous access. The FMC_CLK clock is activated when the CCLKEN is set.
Note: The CCLKEN bit of the FMC_BCR2..4 registers is don’t care. It is only enabled through the FMC_BCR1 register. Bank 1 must be configured in Synchronous mode to generate the FMC_CLK continuous clock.
If CCLKEN bit is set, the FMC_CLK clock ratio is specified by CLKDIV value in the FMC_BTR1 register. CLKDIV in FMC_BWTR1 is don’t care.
If the Synchronous mode is used and CCLKEN bit is set, the synchronous memories connected to other banks than Bank 1 are clocked by the same clock (the CLKDIV value in the FMC_BTR2..4 and FMC_BWTR2..4 registers for other banks has no effect.)

Bit 19 CBURSTRW: Write burst enable
For PSRAM (CRAM) operating in Burst mode, the bit enables synchronous accesses during write operations. The enable bit for synchronous read accesses is the BURSTEN bit in the FMC_BCRx register.
0: Write operations are always performed in Asynchronous mode
1: Write operations are performed in Synchronous mode.

Bits 18:16 CPSIZE[2:0]: CRAM Page Size
These are used for Cellular RAM 1.5 which does not allow burst access to cross the address boundaries between pages. When these bits are configured, the FMC controller splits automatically the burst access when the memory page size is reached (refer to memory datasheet for page size).
000: No burst split when crossing page boundary (default after reset).
001: 128 bytes
010: 256 bytes
100: 1024 bytes
Other configuration: reserved.

Bit 15 ASYNCSRTP: Wait signal during asynchronous transfers
This bit enables/disables the FMC to use the wait signal even during an asynchronous protocol.
0: NWAIT signal is not taken in to account when running an asynchronous protocol (default after reset)
1: NWAIT signal is taken in to account when running an asynchronous protocol

Bit 14 EXTMOD: Extended mode enable.
This bit enables the FMC to program the write timings for asynchronous accesses inside the FMC_BWTR register, thus resulting in different timings for read and write operations.
0: values inside FMC_BWTR register are not taken into account (default after reset)
1: values inside FMC_BWTR register are taken into account

Note: When the Extended mode is disabled, the FMC can operate in Mode1 or Mode2 as follows:
- Mode 1 is the default mode when the SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01)
- Mode 2 is the default mode when the NOR memory type is selected (MTYP = 0x10).
Bit 13 **WAITEN**: Wait enable bit

This bit enables/disables wait-state insertion via the NWAIT signal when accessing the memory in Synchronous mode.

0: NWAIT signal is disabled (its level not taken into account, no wait state inserted after the programmed Flash latency period)
1: NWAIT signal is enabled (its level is taken into account after the programmed latency period to insert wait states if asserted) (default after reset)

Bit 12 **WREN**: Write enable bit

This bit indicates whether write operations are enabled/disabled in the bank by the FMC:

0: Write operations are disabled in the bank by the FMC, an AXI slave error is reported,
1: Write operations are enabled for the bank by the FMC (default after reset).

Bit 11 **WAITCFG**: Wait timing configuration

The NWAIT signal indicates whether the data from the memory are valid or if a wait state must be inserted when accessing the memory in Synchronous mode. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait state or during the wait state:

0: NWAIT signal is active one data cycle before wait state (default after reset),
1: NWAIT signal is active during wait state (not used for PSRAM).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **WAITPOL**: Wait signal polarity bit

This bit defines the polarity of the wait signal from memory used for either in Synchronous or Asynchronous mode:

0: NWAIT active low (default after reset),
1: NWAIT active high.

Bit 8 **BURSTEN**: Burst enable bit

This bit enables/disables synchronous accesses during read operations. It is valid only for synchronous memories operating in Burst mode:

0: Burst mode disabled (default after reset). Read accesses are performed in Asynchronous mode.
1: Burst mode enable. Read accesses are performed in Synchronous mode.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FACCEN**: Flash access enable

This bit enables NOR Flash memory access operations.

0: Corresponding NOR Flash memory access is disabled
1: Corresponding NOR Flash memory access is enabled (default after reset)

Bits 5:4 **MWID[1:0]**: Memory data bus width

Defines the external memory device width, valid for all type of memories.

00: 8 bits
01: 16 bits (default after reset)
10: 32 bits
11: reserved
Bits 3:2  MTYP[1:0]: Memory type
These bits define the type of external memory attached to the corresponding memory bank:
00: SRAM (default after reset for Bank 2...4)
01: PSRAM (CRAM)
10: NOR Flash/OneNAND Flash (default after reset for Bank 1)
11: reserved

Bit 1  MUXEN: Address/data multiplexing enable bit
When this bit is set, the address and data values are multiplexed on the data bus, valid only with NOR and PSRAM memories:
0: Address/Data non-multiplexed
1: Address/Data multiplexed on databus (default after reset)

Bit 0  MBKEN: Memory bank enable bit
This bit enables the memory bank. After reset Bank1 is enabled, all others are disabled.
Accessing a disabled bank causes an ERROR on AXI bus.
0: Corresponding memory bank is disabled
1: Corresponding memory bank is enabled
SRAM/NOR-Flash chip-select timing registers for bank x (FMC_BTRx)

Address offset: 0x04 + 8 * (x - 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, PSRAM and NOR Flash memories. If the EXTMOD bit is set in the FMC_BCRx register, then this register is partitioned for write and read access, that is, 2 registers are available: one to configure read accesses (this register) and one to configure write accesses (FMC_BWTRx registers).

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

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

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

Bits 29:28 ACCMOD[1:0]: Access mode

These bits specify the Asynchronous access modes as shown in the timing diagrams. They are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

00: access mode A
01: access mode B
10: access mode C
11: access mode D

Bits 27:24 DATLAT[3:0]: (see note below bit descriptions): Data latency for synchronous memory

For synchronous access with read/write Burst mode enabled (BURSTEN / CBURSTRW bits set), these bits define the number of memory clock cycles (+2) to issue to the memory before reading/writing the first data:

This timing parameter is not expressed in fmc_ker_ck periods, but in FMC_CLK periods.

For asynchronous access, this value is don’t care.

0000: Data latency of 2 FMC_CLK clock cycles for first burst access
1111: Data latency of 17 FMC_CLK clock cycles for first burst access (default value after reset)

Bits 23:20 CLKDIV[3:0]: Clock divide ratio (for FMC_CLK signal)

These bits define the period of FMC_CLK clock output signal, expressed in number of fmc_ker_ck cycles:

0000: FMC_CLK period = 1 x fmc_ker_ck period
0001: FMC_CLK period = 2 x fmc_ker_ck periods
0010: FMC_CLK period = 3 x fmc_ker_ck periods
1111: FMC_CLK period = 16 x fmc_ker_ck periods (default value after reset)

In asynchronous NOR Flash, SRAM or PSRAM accesses, this value is don’t care.

Note: Refer to Section 23.7.5: Synchronous transactions for FMC_CLK divider ratio formula)
Bits 19:16 **BUSTURN**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of a write-to-read (and read-to-write) transaction. This delay allows to match the minimum time between consecutive transactions \((t_{EHEL} \text{ from } \text{NEx high to } \text{NEx low})\) and the maximum time needed by the memory to free the data bus after a read access \((t_{EHQZ})\). The programmed bus turnaround delay is inserted between an asynchronous read (muxed or mode D) or write transaction and any other asynchronous/synchronous read or write to or from a static bank. The bank can be the same or different in case of read, in case of write the bank can be different except for muxed or mode D.

In some cases, whatever the programmed BUSTURN values, the bus turnaround delay is fixed as follows:

- The bus turnaround delay is not inserted between two consecutive asynchronous write transfers to the same static memory bank except for muxed and D modes.
- There is a bus turnaround delay of 1 FMC clock cycle between:
  - Two consecutive asynchronous read transfers to the same static memory bank except for muxed and D modes.
  - An asynchronous read to an asynchronous or synchronous write to any static bank or dynamic bank except for muxed and D modes.
  - An asynchronous (modes 1, 2, A, B or C) read and a read from another static bank.
- There is a bus turnaround delay of 2 FMC clock cycle between:
  - Two consecutive synchronous writes (burst or single) to the same bank.
  - A synchronous write (burst or single) access and an asynchronous write or read transfer to or from static memory bank (the bank can be the same or different for the case of read).
  - Two consecutive synchronous reads (burst or single) followed by any synchronous/asynchronous read or write from/to another static memory bank.
- There is a bus turnaround delay of 3 FMC clock cycle between:
  - Two consecutive synchronous writes (burst or single) to different static bank.
  - A synchronous write (burst or single) access and a synchronous read from the same or a different bank.

0000: BUSTURN phase duration = 0 fmc_ker_ck clock cycle added

...  
1111: BUSTURN phase duration = 15 x fmc_ker_ck clock cycles added (default value after reset)

Bits 15:8 **DATAST**: Data-phase duration

These bits are written by software to define the duration of the data phase (refer to **Figure 107** to **Figure 119**), used in asynchronous accesses:

0000 0000: Reserved
0000 0001: DATAST phase duration = 1 x fmc_ker_ck clock cycles
0000 0010: DATAST phase duration = 2 x fmc_ker_ck clock cycles

...  
1111 1111: DATAST phase duration = 255 x fmc_ker_ck clock cycles (default value after reset)

For each memory type and access mode data-phase duration, please refer to the respective figure (**Figure 107** to **Figure 119**).

Example: Mode1, write access, DATAST = 1: Data-phase duration = DATAST+1 = 1 x fmc_ker_ck clock cycles.

**Note:** In synchronous accesses, this value is don’t care.
Note: PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to extend the latency as needed. On PSRAMs (CRAMs) the filled DATLAT must be set to 0, so that the FMC exits its latency phase soon and starts sampling NWAIT from memory, then starts to read or write when the memory is ready. This method can be used also with the latest generation of synchronous Flash memories that issue the NWAIT signal, unlike older Flash memories (check the datasheet of the specific Flash memory being used).
SRAM/NOR-Flash write timing registers for bank x (FMC_BWTRx)

Address offset: 0x104 + 8 * (x - 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank. It is used for SRAMs, PSRAMs and NOR Flash memories. When the EXTMOD bit is set in the FMC_BCRx register, then this register is active for write access.

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

| Bits 31:30 | Reserved, must be kept at reset value. |
| Bits 29:28 | ACCMOD: Access mode. |
| These bits specify the asynchronous access modes as shown in the next timing diagrams. These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1. |
| 00: access mode A |
| 01: access mode B |
| 10: access mode C |
| 11: access mode D |
| Bits 27:20 | Reserved, must be kept at reset value. |
| Bits 19:16 | BUSTURN: Bus turnaround phase duration |
| These bits are written by software to add a delay at the end of a write transaction to match the minimum time between consecutive transactions (t_{EH} from ENx high to ENx low): |
| (BUSTURN + 1) fmc_ker_ck period ≥ t_{EHmin} |
| The programmed bus turnaround delay is inserted between an asynchronous write transfer and any other asynchronous/synchronous read or write transfer to or from a static bank. The bank can be the same or different in case of read, in case of write the bank can be different except for muxed or mode D. |
| In some cases, whatever the programmed BUSTURN values, the bus turnaround delay is fixed as follows: |
| • The bus turnaround delay is not inserted between two consecutive asynchronous write transfers to the same static memory bank except for muxed and D modes. |
| • There is a bus turnaround delay of 2 FMC clock cycle between: |
| –Two consecutive synchronous writes (burst or single) to the same bank. |
| –A synchronous write (burst or single) transfer and an asynchronous write or read transfer to or from static memory bank. |
| • There is a bus turnaround delay of 3 FMC clock cycle between: |
| –Two consecutive synchronous writes (burst or single) to different static bank. |
| –A synchronous write (burst or single) transfer and a synchronous read from the same or a different bank. |
| 0000: BUSTURN phase duration = 0 fmc_ker_ck clock cycle added |
| ... |
| 1111: BUSTURN phase duration = 15 fmc_ker_ck clock cycles added (default value after reset) |
Bits 15:8 **DATAST**: Data-phase duration.

These bits are written by software to define the duration of the data phase (refer to Figure 107 to Figure 119), used in asynchronous SRAM, PSRAM and NOR Flash memory accesses:

- **0000 0000**: Reserved
- **0000 0001**: DATAST phase duration = 1 × fmc_ker_ck clock cycles
- **0000 0010**: DATAST phase duration = 2 × fmc_ker_ck clock cycles
  ...
- **1111 1111**: DATAST phase duration = 255 × fmc_ker_ck clock cycles (default value after reset)

Bits 7:4 **ADDHLD**: Address-hold phase duration.

These bits are written by software to define the duration of the address hold phase (refer to Figure 107 to Figure 119), used in asynchronous multiplexed accesses:

- **0000**: Reserved
- **0001**: ADDHLD phase duration = 1 × fmc_ker_ck clock cycle
- **0010**: ADDHLD phase duration = 2 × fmc_ker_ck clock cycle
  ...
- **1111**: ADDHLD phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)

*Note: In synchronous NOR Flash accesses, this value is not used, the address hold phase is always 1 Flash clock period duration.*

Bits 3:0 **ADDSET**: Address setup phase duration.

These bits are written by software to define the duration of the address setup phase in fmc_ker_ck cycles (refer to Figure 107 to Figure 119), used in asynchronous accesses:

- **0000**: ADDSET phase duration = 0 × fmc_ker_ck clock cycle
  ...
- **1111**: ADDSET phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)

*Note: In synchronous accesses, this value is not used, the address setup phase is always 1 Flash clock period duration. In muxed mode, the minimum ADDSET value is 1.*
23.8 NAND Flash controller

The FMC generates the appropriate signal timings to drive 8- and 16-bit NAND Flash memories.

The NAND bank is configured through dedicated registers (Section 23.8.7). The programmable memory parameters include access timings (shown in Table 170) and ECC configuration.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Access mode</th>
<th>Unit</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory setup time</td>
<td>Number of clock cycles (fmc_ker_ck) required to set up the address before the command assertion</td>
<td>Read/Write</td>
<td>AHB clock cycle (fmc_ker_ck)</td>
<td>1</td>
<td>255</td>
</tr>
<tr>
<td>Memory wait</td>
<td>Minimum duration (in fmc_ker_ck clock cycles) of the command assertion</td>
<td>Read/Write</td>
<td>AHB clock cycle (fmc_ker_ck)</td>
<td>2</td>
<td>255</td>
</tr>
<tr>
<td>Memory hold</td>
<td>Number of clock cycles (fmc_ker_ck) during which the address must be held (as well as the data if a write access is performed) after the command deassertion</td>
<td>Read/Write</td>
<td>AHB clock cycle (fmc_ker_ck)</td>
<td>1</td>
<td>254</td>
</tr>
<tr>
<td>Memory databus high-Z</td>
<td>Number of clock cycles (fmc_ker_ck) during which the data bus is kept in high-Z state after a write access has started</td>
<td>Write</td>
<td>AHB clock cycle (fmc_ker_ck)</td>
<td>0</td>
<td>254</td>
</tr>
</tbody>
</table>

23.8.1 External memory interface signals

The following tables list the signals that are typically used to interface NAND Flash memories.

Note: The prefix "N" identifies the signals which are active low.

8-bit NAND Flash memory

<table>
<thead>
<tr>
<th>FMC pin name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[17]</td>
<td>O</td>
<td>NAND Flash address latch enable (ALE) signal</td>
</tr>
<tr>
<td>A[16]</td>
<td>O</td>
<td>NAND Flash command latch enable (CLE) signal</td>
</tr>
<tr>
<td>D[7:0]</td>
<td>I/O</td>
<td>8-bit multiplexed, bidirectional address/data bus</td>
</tr>
<tr>
<td>NCE</td>
<td>O</td>
<td>Chip Select</td>
</tr>
<tr>
<td>NOE (= NRE)</td>
<td>O</td>
<td>Output enable (memory signal name: read enable, NRE)</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NWAIT/INT</td>
<td>I</td>
<td>NAND Flash ready/busy input signal to the FMC</td>
</tr>
</tbody>
</table>
Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

### 16-bit NAND Flash memory

#### Table 172. 16-bit NAND Flash memory

<table>
<thead>
<tr>
<th>FMC pin name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[17]</td>
<td>O</td>
<td>NAND Flash address latch enable (ALE) signal</td>
</tr>
<tr>
<td>A[16]</td>
<td>O</td>
<td>NAND Flash command latch enable (CLE) signal</td>
</tr>
<tr>
<td>D[15:0]</td>
<td>I/O</td>
<td>16-bit multiplexed, bidirectional address/data bus</td>
</tr>
<tr>
<td>NCE</td>
<td>O</td>
<td>Chip Select</td>
</tr>
<tr>
<td>NOE(= NRE)</td>
<td>O</td>
<td>Output enable (memory signal name: read enable, NRE)</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NWAIT/INT</td>
<td>I</td>
<td>NAND Flash ready/busy input signal to the FMC</td>
</tr>
</tbody>
</table>

**Note:** Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

### 23.8.2 NAND Flash supported memories and transactions

*Table 173 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND Flash controller are shown in gray.*

#### Table 173. Supported memories and transactions

<table>
<thead>
<tr>
<th>Device</th>
<th>Mode</th>
<th>R/W</th>
<th>AXI data size</th>
<th>Memory data size</th>
<th>Allowed/ not allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAND 8-bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asynchronous R</td>
<td>R</td>
<td>8</td>
<td>8</td>
<td>Y</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Asynchronous W</td>
<td>W</td>
<td>8</td>
<td>8</td>
<td>Y</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Asynchronous R</td>
<td>R</td>
<td>16</td>
<td>8</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
<td></td>
</tr>
<tr>
<td>Asynchronous W</td>
<td>W</td>
<td>16</td>
<td>8</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
<td></td>
</tr>
<tr>
<td>Asynchronous R</td>
<td>R</td>
<td>32</td>
<td>8</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
<td></td>
</tr>
<tr>
<td>Asynchronous W</td>
<td>W</td>
<td>32</td>
<td>8</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
<td></td>
</tr>
<tr>
<td>Asynchronous R</td>
<td>R</td>
<td>32</td>
<td>8</td>
<td>Y</td>
<td>Split into 8 FMC accesses</td>
<td></td>
</tr>
<tr>
<td>Asynchronous W</td>
<td>W</td>
<td>32</td>
<td>8</td>
<td>Y</td>
<td>Split into 8 FMC accesses</td>
<td></td>
</tr>
</tbody>
</table>
23.8.3 Timing diagrams for NAND Flash memories

The NAND Flash memory bank is managed through a set of registers:

- Control register: FMC_PCR
- Interrupt status register: FMC_SR
- ECC register: FMC_ECCR
- Timing register for Common memory space: FMC_PMEM
- Timing register for Attribute memory space: FMC_PATT

Each timing configuration register contains three parameters used to define the number of fmc_ker_ck cycles for the three phases of any NAND Flash access, plus one parameter that defines the timing to start driving the data bus when a write access is performed. Figure 125 shows the timing parameter definitions for common memory accesses, knowing that Attribute memory space access timings are similar.
23.8.4 NAND Flash operations

The command latch enable (CLE) and address latch enable (ALE) signals of the NAND Flash memory device are driven by address signals from the FMC controller. This means that to send a command or an address to the NAND Flash memory, the CPU has to perform a write to a specific address in its memory space.

A typical page read operation from the NAND Flash device requires the following steps:

1. Program and enable the corresponding memory bank by configuring the FMC_PCR and FMC_PMEM (and for some devices, FMC_PATT, see Section 23.8.5: NAND Flash prewait feature) registers according to the characteristics of the NAND Flash memory (PWID bits for the data bus width of the NAND Flash memory, PWAITEN = 0 or 1 as needed, see Section 23.6.2: NAND Flash memory address mapping for timing configuration).

2. The CPU performs a byte write to the common memory space, with data byte equal to one Flash command byte (for example 0x00 for Samsung NAND Flash devices). The LE input of the NAND Flash memory is active during the write strobe (low pulse on NWE), thus the written byte is interpreted as a command by the NAND Flash memory. Once the command is latched by the memory device, it does not need to be written again for the following page read operations.

3. The CPU can send the start address (STARTAD) for a read operation by writing four bytes (or three for smaller capacity devices), STARTAD[7:0], STARTAD[16:9], STARTAD[24:17] and finally STARTAD[25] (for 64 Mb x 8 bit NAND Flash memories) in the common memory or attribute space. The ALE input of the NAND Flash device is active during the write strobe (low pulse on NWE), thus the written bytes are interpreted as the start address for read operations. Using the attribute memory space makes it possible to use a different timing configuration of the FMC, which can be used.
to implement the prewait functionality needed by some NAND Flash memories (see details in Section 23.8.5: NAND Flash prewait feature).

4. The controller waits for the NAND Flash memory to be ready (R/NB signal high), before starting a new access to the same or another memory bank. While waiting, the controller holds the NCE signal active (low).

5. The CPU can then perform byte read operations from the common memory space to read the NAND Flash page (data field + Spare field) byte by byte.

6. The next NAND Flash page can be read without any CPU command or address write operation. This can be done in three different ways:
   - by simply performing the operation described in step 5
   - a new random address can be accessed by restarting the operation at step 3
   - a new command can be sent to the NAND Flash device by restarting at step 2

23.8.5 NAND Flash prewait feature

Some NAND Flash devices require that, after writing the last part of the address, the controller waits for the R/NB signal to go low. (see Figure 126).

![Figure 126. Access to non 'CE don't care' NAND-Flash](image-url)

1. CPU wrote byte 0x00 at address 0x7001 0000.
2. CPU wrote byte A7–A0 at address 0x7002 0000.
3. CPU wrote byte A16–A9 at address 0x7002 0000.
4. CPU wrote byte A24–A17 at address 0x7002 0000.
5. CPU wrote byte A25 at address 0x8802 0000; FMC performs a write access using FMC_PATT2 timing definition, where ATTHOLD ≥ 7 (providing that (7+1) × fmc_ker_ck = 112 ns > tWB max). This guarantees that NCE remains low until R/NB goes low and high again (only requested for NAND Flash memories where NCE is not don't care).
When this function is required, it can be performed by programming the MEMHOLD value to meet the t\textsubscript{WB} timing. However, any CPU read access to NAND Flash memory has a hold delay of (MEMHOLD + 1) \text{fmc\_ker\_ck} cycles, and any CPU write access has a hold delay of (MEMHOLD) \text{fmc\_ker\_ck} cycles that is inserted between the rising edge of the NWE signal and the next access.

To cope with this timing constraint, the attribute memory space can be used by programming its timing register with an ATTHOLD value that meets the t\textsubscript{WB} timing, and by keeping the MEMHOLD value at its minimum value. The CPU must then use the common memory space for all NAND Flash read and write accesses, except when writing the last address byte to the NAND Flash device, where the CPU must write to the attribute memory space.

23.8.6 **Computation of the error correction code (ECC) in NAND Flash memory**

The FMC controller includes an error correction code computation hardware block. It reduces the host CPU workload when processing the ECC by software. The ECC block is associated with NAND bank.

The ECC algorithm implemented in the FMC can perform 1-bit error correction and 2-bit error detection per 256, 512, 1024, 2048, 4096 or 8192 bytes read or written from/to the NAND Flash memory. It is based on the Hamming coding algorithm and consists in calculating the row and column parity.

The ECC modules monitor the NAND Flash data bus and read/write signals (NCE and NWE) each time the NAND Flash memory bank is active.

The ECC operates as follows:

- When accessing NAND Flash bank, the data present on the D[15:0] bus is latched and used for ECC computation.
- When accessing any other address in NAND Flash memory, the ECC logic is idle, and does not perform any operation. As a result, write operations to define commands or addresses to the NAND Flash memory are not taken into account for ECC computation.

Once the desired number of bytes has been read/written from/to the NAND Flash memory by the host CPU, the FMC\_ECCR registers must be read to retrieve the computed value. Once read, they should be cleared by resetting the ECCEN bit to ‘0’. To compute a new data block, the ECCEN bit must be set to one in the FMC\_PCR registers.
Execute below the sequence to perform an ECC computation:
1. Enable the ECCEN bit in the FMC_PCR register.
2. Write data to the NAND Flash memory page. While the NAND page is written, the ECC block computes the ECC value.
3. Wait until the ECC code is ready (FIFO empty).
4. Read the ECC value available in the FMC_ECCR register and store it in a variable.
5. Clear the ECCEN bit and then enable it in the FMC_PCR register before reading back the written data from the NAND page. While the NAND page is read, the ECC block computes the ECC value.
6. Read the new ECC value available in the FMC_ECCR register.
7. If the two ECC values are the same, no correction is required, otherwise there is an ECC error and the software correction routine returns information on whether the error can be corrected or not.

23.8.7 NAND Flash controller registers

NAND Flash control registers (FMC_PCR)
Address offset: 0x80
Reset value: 0x0000 0018

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

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAR2</td>
<td>TAR1</td>
<td>TAR0</td>
<td>TCLR</td>
<td>CESP</td>
<td>PWID</td>
<td>PW2</td>
<td>PBKEN</td>
<td>PWAITEN</td>
<td>P3</td>
<td>P2</td>
<td>P1</td>
<td>P0</td>
<td>P9</td>
<td>P8</td>
<td>P7</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:20  Reserved, must be kept at reset value.

Bits 19:17  **ECCPS**: ECC page size.
These bits define the page size for the extended ECC:
- 000: 256 bytes
- 001: 512 bytes
- 010: 1024 bytes
- 011: 2048 bytes
- 100: 4096 bytes
- 101: 8192 bytes

Bits 16:13  **TAR**: ALE to RE delay.
These bits set time from ALE low to RE low in number of fmc_ker_ck clock cycles.
Time is: \( t_{ar} = (TAR + SET + 2) \times t_{fmc_ker_ck} \) where \( t_{fmc_ker_ck} \) is the FMC clock period
- 0000: 1 x fmc_ker_ck cycle (default)
- 1111: 16 x fmc_ker_ck cycles

*Note: Set is MEMSET or ATTSET according to the addressed space.*
Bits 12:9 **TCLR**: CLE to RE delay.
These bits set time from CLE low to RE low in number of fmc_ker_ck clock cycles. The time is given by the following formula:
\[ t_{clr} = (TCLR + SET + 2) \times t_{fmc_ker_ck} \]
where \( t_{fmc_ker_ck} \) is the fmc_ker_ck clock period
0000: 1 \times fmc_ker_ck cycle (default)
1111: 16 \times fmc_ker_ck cycles

---

**Note**: Set is MEMSET or ATTSET according to the addressed space.

Bits 8:7 Reserved, must be kept at reset value.

**Bit 6 ECCEN**: ECC computation logic enable bit
0: ECC logic is disabled and reset (default after reset),
1: ECC logic is enabled.

**Bits 5:4 PWID**: Data bus width.
These bits define the external memory device width.
00: 8 bits
01: 16 bits (default after reset).
10: reserved.
11: reserved.

**Bit 3** Reserved, must be kept at reset value.

**Bit 2 PBKEN**: NAND Flash memory bank enable bit.
This bit enables the memory bank. Accessing a disabled memory bank causes an ERROR on AXI bus
0: Corresponding memory bank is disabled (default after reset)
1: Corresponding memory bank is enabled

**Bit 1 PWAITEN**: Wait feature enable bit.
This bit enables the Wait feature for the NAND Flash memory bank:
0: disabled
1: enabled

**Bit 0** Reserved, must be kept at reset value.

---

**FIFO status and interrupt register (FMC_SR)**

**Address offset**: 0x84

**Reset value**: 0x0000 0040

This register contains information about the FIFO status and interrupt. The FMC features a FIFO that is used when writing to memories to transfer up to 16 words of data.
This is used to quickly write to the FIFO and free the AXI bus for transactions to peripherals other than the FMC, while the FMC is draining its FIFO into the memory. One of these register bits indicates the status of the FIFO, for ECC purposes.

The ECC is calculated while the data are written to the memory. To read the correct ECC, the software must consequently wait until the FIFO is empty.
Bits 31:7  Reserved, must be kept at reset value.

Bit 6  **FEMPT:** FIFO empty.
Read-only bit that provides the status of the FIFO
0: FIFO not empty
1: FIFO empty

Bit 5  **IFEN:** Interrupt falling edge detection enable bit
0: Interrupt falling edge detection request disabled
1: Interrupt falling edge detection request enabled

Bit 4  **ILEN:** Interrupt high-level detection enable bit
0: Interrupt high-level detection request disabled
1: Interrupt high-level detection request enabled

Bit 3  **IREN:** Interrupt rising edge detection enable bit
0: Interrupt rising edge detection request disabled
1: Interrupt rising edge detection request enabled

Bit 2  **IFS:** Interrupt falling edge status
The flag is set by hardware and reset by software.
0: No interrupt falling edge occurred
1: Interrupt falling edge occurred

*Note: If this bit is written by software to 1 it will be set.*

Bit 1  **ILS:** Interrupt high-level status
The flag is set by hardware and reset by software.
0: No Interrupt high-level occurred
1: Interrupt high-level occurred

Bit 0  **IRS:** Interrupt rising edge status
The flag is set by hardware and reset by software.
0: No interrupt rising edge occurred
1: Interrupt rising edge occurred

*Note: If this bit is written by software to 1 it will be set.*
Common memory space timing register (FMC_PMEM)

Address offset: Address: 0x88
Reset value: 0xFCFC FCFC

The FMC_PMEM read/write register contains the timing information for NAND Flash memory bank. This information is used to access either the common memory space of the NAND Flash for command, address write access and data read/write access.

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

Bits 31:24 MEMHIZ: Common memory x data bus Hi-Z time
These bits define the number of fmc_ker_ck clock cycles during which the data bus is kept Hi-Z after the start of a NAND Flash write access to common memory space. This is only valid for write transactions:
0000 0000: 0 x fmc_ker_ck cycle
1111 1110: 254 x fmc_ker_ck cycles
1111 1111: reserved.

Bits 23:16 MEMHOLD: Common memory hold time
These bits define the number of fmc_ker_ck clock cycles for write accesses and fmc_ker_ck+1 clock cycles for read accesses during which the address is held (and data for write accesses) after the command is deasserted (NWE, NOE), for NAND Flash read or write access to common memory space:
0000 0000: reserved.
0000 0001: 1 fmc_ker_ck cycle for write access / 3 fmc_ker_ck cycle for read access
1111 1110: 254 fmc_ker_ck cycles for write access / 257 fmc_ker_ck cycles for read access
1111 1111: reserved.

Bits 15:8 MEMWAIT: Common memory wait time
These bits define the minimum number of fmc_ker_ck (+1) clock cycles to assert the command (NWE, NOE), for NAND Flash read or write access to common memory space. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of fmc_ker_ck:
0000 0000: reserved
0000 0001: x fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)
1111 1110: 255 x fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)
1111 1111: reserved.

Bits 7:0 MEMSET: Common memory x setup time
These bits define the number of fmc_ker_ck (+1) clock cycles to set up the address before the command assertion (NWE, NOE), for NAND Flash read or write access to common memory space:
0000 0000: fmc_ker_ck cycles
1111 1110: 255 x fmc_ker_ck cycles
1111 1111: reserved
**Attribute memory space timing registers (FMC_PATT)**

Address offset: 0x8C

Reset value: 0xFCFC FCFC

The FMC_PATT read/write register contains the timing information for NAND Flash memory bank. It is used for 8-bit accesses to the attribute memory space of the NAND Flash for the last address write access if the timing must differ from that of previous accesses (for Ready/Busy management, refer to Section 23.8.5: NAND Flash prewait feature).

<table>
<thead>
<tr>
<th></th>
<th>ATTHIZ</th>
<th>ATTHOLD</th>
<th>ATTWAIT</th>
<th>ATTSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ATTHIZ</strong>: Attribute memory data bus Hi-Z time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>These bits define the number of fmc_ker_ck clock cycles during which the data bus is kept in Hi-Z after the start of a NAND Flash write access to attribute memory space on socket. Only valid for writ transaction:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000 0000: 0 x fmc_ker_ck cycle</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111 1110: 254 x fmc_ker_ck cycles</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111 1111: reserved.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Bits 23:16 |         |         |         |        |
| **ATTHOLD**: Attribute memory hold time |         |         |         |        |
| These bits define the number of fmc_ker_ck clock cycles during which the address is held (and data for write access) after the command deassertion (NWE, NOE), for NAND Flash read or write access to attribute memory space: |         |         |         |        |
| 0000 0000: reserved |         |         |         |        |
| 0000 0001: 1 x fmc_ker_ck cycle |         |         |         |        |
| 1111 1110: 254 x fmc_ker_ck cycles |         |         |         |        |
| 1111 1111: reserved. |         |         |         |        |

| Bits 15:8 |         |         |         |        |
| **ATTWAIT**: Attribute memory wait time |         |         |         |        |
| These bits define the minimum number of x fmc_ker_ck (+1) clock cycles to assert the command (NWE, NOE), for NAND Flash read or write access to attribute memory space. The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of fmc_ker_ck: |         |         |         |        |
| 0000 0000: reserved |         |         |         |        |
| 0000 0001: 2 x fmc_ker_ck cycles (+ wait cycle introduced by deassertion of NWAIT) |         |         |         |        |
| 1111 1110: 255 x fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT) |         |         |         |        |
| 1111 1111: reserved. |         |         |         |        |

| Bits 7:0 |         |         |         |        |
| **ATTSET**: Attribute memory setup time |         |         |         |        |
| These bits define the number of fmc_ker_ck (+1) clock cycles to set up address before the command assertion (NWE, NOE), for NAND Flash read or write access to attribute memory space: |         |         |         |        |
| 0000 0000: 1 x fmc_ker_ck cycle |         |         |         |        |
| 1111 1110: 255 x fmc_ker_ck cycles |         |         |         |        |
| 1111 1111: reserved. |         |         |         |        |
ECC result registers (FMC_ECCR)

Address offset: 0x94
Reset value: 0x0000 0000

This register contain the current error correction code value computed by the ECC computation modules of the FMC NAND controller. When the CPU reads/writes the data from a NAND Flash memory page at the correct address (refer to Section 23.8.6: Computation of the error correction code (ECC) in NAND Flash memory), the data read/written from/to the NAND Flash memory are processed automatically by the ECC computation module. When X bytes have been read (according to the ECCPS field in the FMC_PCR registers), the CPU must read the computed ECC value from the FMC_ECC registers. It then verifies if these computed parity data are the same as the parity value recorded in the spare area, to determine whether a page is valid, and, to correct it otherwise. The FMC_ECCR register should be cleared after being read by setting the ECCEN bit to ’0’. To compute a new data block, the ECCEN bit must be set to ’1’.

<table>
<thead>
<tr>
<th>ECC[31:16]</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>31-16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ECC[15:0]</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **ECC[31:0]: ECC result**
This field contains the value computed by the ECC computation logic. **Table 174** describes the contents of these bitfields.

**Table 174. ECC result relevant bits**

<table>
<thead>
<tr>
<th>ECCPS[2:0]</th>
<th>Page size in bytes</th>
<th>ECC bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>256</td>
<td>ECC[21:0]</td>
</tr>
<tr>
<td>001</td>
<td>512</td>
<td>ECC[23:0]</td>
</tr>
<tr>
<td>010</td>
<td>1024</td>
<td>ECC[25:0]</td>
</tr>
<tr>
<td>011</td>
<td>2048</td>
<td>ECC[27:0]</td>
</tr>
<tr>
<td>100</td>
<td>4096</td>
<td>ECC[29:0]</td>
</tr>
<tr>
<td>101</td>
<td>8192</td>
<td>ECC[31:0]</td>
</tr>
</tbody>
</table>
23.9 SDRAM controller

23.9.1 SDRAM controller main features

The main features of the SDRAM controller are the following:

- Two SDRAM banks with independent configuration
- 8-bit, 16-bit, 32-bit data bus width
- 13-bits Address Row, 11-bits Address Column, 4 internal banks: 4x16Mx32bit (256 MB), 4x16Mx16bit (128 MB), 4x16Mx8bit (64 MB)
- Word, half-word, byte access
- SDRAM clock can be fmc_ker_ck/2 or fmc_ker_ck/3
- Automatic row and bank boundary management
- Multibank ping-pong access
- Programmable timing parameters
- Automatic Refresh operation with programmable Refresh rate
- Self-refresh mode
- Power-down mode
- SDRAM power-up initialization by software
- CAS latency of 1,2,3
- Cacheable Read FIFO with depth of 6 lines x32-bit (6 x14-bit address tag)

23.9.2 SDRAM External memory interface signals

At startup, the SDRAM I/O pins used to interface the FMC SDRAM controller with the external SDRAM devices must configured by the user application. The SDRAM controller I/O pins which are not used by the application, can be used for other purposes.

<table>
<thead>
<tr>
<th>SDRAM signal</th>
<th>I/O type</th>
<th>Description</th>
<th>Alternate function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDCLK</td>
<td>O</td>
<td>SDRAM clock</td>
<td>-</td>
</tr>
<tr>
<td>SDCKE[1:0]</td>
<td>O</td>
<td>SDCKE0: SDRAM Bank 1 Clock Enable SDCKE1: SDRAM Bank 2 Clock Enable</td>
<td>-</td>
</tr>
<tr>
<td>SDNE[1:0]</td>
<td>O</td>
<td>SDNE0: SDRAM Bank 1 Chip Enable SDNE1: SDRAM Bank 2 Chip Enable</td>
<td>-</td>
</tr>
<tr>
<td>D[31:0]</td>
<td>I/O</td>
<td>Bidirectional data bus</td>
<td>FMC_D[31:0]</td>
</tr>
<tr>
<td>BA[1:0]</td>
<td>O</td>
<td>Bank Address</td>
<td>FMC_A[15:14]</td>
</tr>
<tr>
<td>NRAS</td>
<td>O</td>
<td>Row Address Strobe</td>
<td>-</td>
</tr>
<tr>
<td>NCAS</td>
<td>O</td>
<td>Column Address Strobe</td>
<td>-</td>
</tr>
<tr>
<td>SDNWE</td>
<td>O</td>
<td>Write Enable</td>
<td>-</td>
</tr>
<tr>
<td>NBL[3:0]</td>
<td>O</td>
<td>Output Byte Mask for write accesses (memory signal name: DQM[3:0])</td>
<td>FMC_NBL[3:0]</td>
</tr>
</tbody>
</table>
23.9.3 **SDRAM controller functional description**

All SDRAM controller outputs (signals, address and data) change on the falling edge of the memory clock (FMC_SDCLK).

**SDRAM initialization**

The initialization sequence is managed by software. If the two banks are used, the initialization sequence must be generated simultaneously to Bank 1 and Bank 2 by setting the Target Bank bits CTB1 and CTB2 in the FMC_SDCMR register:

1. Program the memory device features into the FMC_SDCRx register. The SDRAM clock frequency, RBURST and RPIPE must be programmed in the FMC_SDCR1 register.
2. Program the memory device timing into the FMC_SDTRx register. The TRP and TRC timings must be programmed in the FMC_SDTR1 register.
3. Set MODE bits to '001' and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to start delivering the clock to the memory (SDCKE is driven high).
4. Wait during the prescribed delay period. Typical delay is around 100 μs (refer to the SDRAM datasheet for the required delay after power-up).
5. Set MODE bits to ‘010’ and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to issue a “Precharge All” command.
6. Set MODE bits to ‘011’, and configure the Target Bank bits (CTB1 and/or CTB2) as well as the number of consecutive Auto-refresh commands (NRFS) in the FMC_SDCMR register. Refer to the SDRAM datasheet for the number of Auto-refresh commands that should be issued. Typical number is 8.
7. Configure the MRD field, set the MODE bits to ‘100’, and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to issue a “Load Mode Register” command and program the SDRAM device. In particular the Burst Length (BL) has to be set to ‘1’) and the CAS latency has to be selected. If the Mode Register is not the same for both SDRAM banks, this step has to be repeated twice, once for each bank and the Target Bank bits set accordingly. For mobile SDRAM devices, the MRD field is also used to configure the extended mode register while issuing the Load Mode Register.
8. Program the refresh rate in the FMC_SDRTR register
   The refresh rate corresponds to the delay between refresh cycles. Its value must be adapted to SDRAM devices.

At this stage the SDRAM device is ready to accept commands. If a system reset occurs during an ongoing SDRAM access, the data bus might still be driven by the SDRAM device. Therefore the SDRAM device must be first reinitialized after reset before issuing any new access by the NOR Flash/PSRAM/SRAM or NAND Flash controller.

**Note:** If two SDRAM devices are connected to the FMC, all the accesses performed at the same time to both devices by the Command Mode register (Load Mode Register command) are issued using the timing parameters configured for SDRAM Bank 1 (TMRD and TRAS timings) in the FMC_SDTR1 register.
**SDRAM controller write cycle**

The SDRAM controller accepts single and burst write requests and translates them into single memory accesses. In both cases, the SDRAM controller keeps track of the active row for each bank to be able to perform consecutive write accesses to different banks (Multibank ping-pong access).

Before performing any write access, the SDRAM bank write protection must be disabled by clearing the WP bit in the FMC_SDCRx register.

![Figure 127. Burst write SDRAM access waveforms](image)

The SDRAM controller always checks the next access.  
- If the next access is in the same row or in another active row, the write operation is carried out,  
- if the next access targets another row (not active), the SDRAM controller generates a precharge command, activates the new row and initiates a write command.
SDRAM controller read cycle

The SDRAM controller accepts single and burst read requests and translates them into single memory accesses. In both cases, the SDRAM controller keeps track of the active row in each bank to be able to perform consecutive read accesses in different banks (Multibank ping-pong access).

**Figure 128. Burst read SDRAM access**

The FMC SDRAM controller features a Cacheable read FIFO (6 lines x 32 bits). It is used to store data read in advance during the CAS latency period (up to 3 memory clock cycles, programmed FMC_SDCRx) and during the RPIPE delay when set to 2xfmc_ker_ck clock cycles as configured in FMC_SDCR1) following this formula: CAS Latency + 1 + (RPIPE DIV2). The RBURST bit must be set in the FMC_SDCR1 register to anticipate the next read access.

**Examples:**
- CAS=3, RPIPE= 2xfmc_ker_ck. In this case, 5 data (not committed) are stored in the FIFO (4 data during CAS latency and 1 data during RPIPE delay)
- CAS=3, RPIPE= 1xfmc_ker_ck. In this case, 4 data (not committed) are stored in the FIFO (4 data during CAS latency)

The read FIFO features a 14-bit address tag to each line to identify its content: 11 bits for the column address, 2 bits to select the internal bank and the active row, and 1 bit to select the SDRAM device.

When the end of the row is reached in advance during an burst read transaction, the data read in advance (not committed) are not stored in the read FIFO. For single read access, data are correctly stored in the FIFO.
Each time a read request occurs, the SDRAM controller checks:

- If the address matches one of the address tags, data are directly read from the FIFO and the corresponding address tag/line content is cleared and the remaining data in the FIFO are compacted to avoid empty lines.
- Otherwise, a new read command is issued to the memory and the FIFO is updated with new data. If the FIFO is full, the older data are lost.

**Figure 129. Logic diagram of Read access with RBURST bit set (CAS=2, RPIPE=0)**

During a write access or a Precharge command, the read FIFO is flushed and ready to be filled with new data.

After the first read request, if the current access was not performed to a row boundary, the SDRAM controller anticipates the next read access during the CAS latency period and the RPIPE delay (if configured). This is done by incrementing the memory address. The following condition must be met:

- RBURST control bit should be set to ‘1’ in the FMC_SDCR1 register.
The address management depends on the next AXI request:

- Next request is sequential (Burst access)
  In this case, the SDRAM controller increments the address.
- Next request is not sequential
  – If the new read request targets the same row or another active row, the new address is passed to the memory and the master is stalled for the CAS latency period, waiting for the new data from memory.
  – If the new read request does not target an active row, the SDRAM controller generates a Precharge command, activates the new row, and initiates a read command.

If the RBURST is reset, the read FIFO is not used.

**Row and bank boundary management**

When a read or write access crosses a row boundary, if the next read or write access is sequential and the current access was performed to a row boundary, the SDRAM controller executes the following operations:

1. Precharge of the active row,
2. Activation of the new row
3. Start of a read/write command.

At a row boundary, the automatic activation of the next row is supported for all columns and data bus width configurations.

If necessary, the SDRAM controller inserts additional clock cycles between the following commands:

- Between Precharge and Active commands to match TRP parameter (only if the next access is in a different row in the same bank),
- Between Active and Read commands to match the TRCD parameter.

These parameters are defined into the FMC_SDTRx register.

Refer to *Figure 127* and *Figure 128* for read and burst write access crossing a row boundary.
Figure 130. Read access crossing row boundary

Figure 131. Write access crossing row boundary
If the next access is sequential and the current access crosses a bank boundary, the SDRAM controller activates the first row in the next bank and initiates a new read/write command. Two cases are possible:

- If the current bank is not the last one, the active row in the new bank must be precharged. At a bank boundary, the automatic activation of the next row is supported for all rows/columns and data bus width configuration.
- If the current bank is the last one, the automatic activation of the next row is supported only when addressing 13-bit rows, 11-bit columns, 4 internal banks and 32-bit data bus SDRAM devices. Otherwise, the SDRAM address range is violated and an AXI slave error is generated.
- In case of 13-bit row address, 11-bit column address, 4 internal banks and bus width 32-bit SDRAM memories, at boundary bank, the SDRAM controller continues to read/write from the second SDRAM device (assuming it has been initialized):
  a) The SDRAM controller activates the first row (after precharging the active row, if there is already an active row in the first internal bank, and initiates a new read/write command.
  b) If the first row is already activated, the SDRAM controller just initiates a read/write command.

**SDRAM controller refresh cycle**

The Auto-refresh command is used to refresh the SDRAM device content. The SDRAM controller periodically issues auto-refresh commands. An internal counter is loaded with the COUNT value in the register FMC_SDRTR. This value defines the number of memory clock cycles between the refresh cycles (refresh rate). When this counter reaches zero, an internal pulse is generated.

If a memory access is ongoing, the auto-refresh request is delayed. However, if the memory access and the auto-refresh requests are generated simultaneously, the auto-refresh request takes precedence.

If the memory access occurs during an auto-refresh operation, the request is buffered and processed when the auto-refresh is complete.

If a new auto-refresh request occurs while the previous one was not served, the RE (Refresh Error) bit is set in the Status register. An Interrupt is generated if it has been enabled (REIE = ‘1’).

If SDRAM lines are not in idle state (not all row are closed), the SDRAM controller generates a PALL (Precharge ALL) command before the auto-refresh.

If the Auto-refresh command is generated by the FMC_SDCMR Command Mode register (Mode bits = ‘011’), a PALL command (Mode bits = ‘010’) must be issued first.
23.9.4 Low-power modes

Two low-power modes are available:

- Self-refresh mode
  The auto-refresh cycles are performed by the SDRAM device itself to retain data without external clocking.
- Power-down mode
  The auto-refresh cycles are performed by the SDRAM controller.

Self-refresh mode

This mode is selected by setting the MODE bits to ‘101’ and by configuring the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register.

The SDRAM clock stops running after a TRAS delay and the internal refresh timer stops counting only if one of the following conditions is met:

- A Self-refresh command is issued to both devices
- One of the devices is not activated (SDRAM bank is not initialized).

Before entering Self-Refresh mode, the SDRAM controller automatically issues a PALL command.

If the Write data FIFO is not empty, all data are sent to the memory before activating the Self-refresh mode and the BUSY status flag remains set.

In Self-refresh mode, all SDRAM device inputs become don’t care except for SDCKE which remains low.

The SDRAM device must remain in Self-refresh mode for a minimum period of time of TRAS and can remain in Self-refresh mode for an indefinite period beyond that. To guarantee this minimum period, the BUSY status flag remains high after the Self-refresh activation during a TRAS delay.

As soon as an SDRAM device is selected, the SDRAM controller generates a sequence of commands to exit from Self-refresh mode. After the memory access, the selected device remains in Normal mode.

To exit from Self-refresh, the MODE bits must be set to ‘000’ (Normal mode) and the Target Bank bits (CTB1 and/or CTB2) must be configured in the FMC_SDCMR register.
Figure 132. Self-refresh mode

Precharge all active banks
Enter Self-refresh mode
CLK stable prior to existing Self-refresh mode
Exit Self-refresh mode (restart refresh timebase)
Power-down mode

This mode is selected by setting the MODE bits to ‘110’ and by configuring the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register.

Figure 133. Power-down mode

If the Write data FIFO is not empty, all data are sent to the memory before activating the Power-down mode.

As soon as an SDRAM device is selected, the SDRAM controller exits from the Power-down mode. After the memory access, the selected SDRAM device remains in Normal mode.

During Power-down mode, all SDRAM device input and output buffers are deactivated except for the SDCKE which remains low.

The SDRAM device cannot remain in Power-down mode longer than the refresh period and cannot perform the Auto-refresh cycles by itself. Therefore, the SDRAM controller carries out the refresh operation by executing the operations below:

1. Exit from Power-down mode and drive the SDCKE high
2. Generate the PALL command only if a row was active during Power-down mode
3. Generate the auto-refresh command
4. Drive SDCKE low again to return to Power-down mode.

To exit from Power-down mode, the MODE bits must be set to ‘000’ (Normal mode) and the Target Bank bits (CTB1 and/or CTB2) must be configured in the FMC_SDCMR register.
### 23.9.5 SDRAM controller registers

**SDRAM Control registers for SDRAM memory bank x (FMC_SDCRx)**

Address offset: 0x140 + 4\(^x\) (x - 1), (x = 1 to 2)

Reset value: 0x0000 02D0

This register contains the control parameters for each SDRAM memory bank

| Bit 31:15 | Reserved, must be kept at reset value. |
| Bit 14:13 | **RPIPE[1:0]: Read pipe**
| | These bits define the delay, in fmc_ker_ck clock cycles, for reading data after CAS latency.
| | 00: No fmc_ker_ck clock cycle delay
| | 01: One fmc_ker_ck clock cycle delay
| | 10: Two fmc_ker_ck clock cycle delay
| | 11: reserved.
| | **Note:** The corresponding bits in the FMC_SDCR2 register is read only.
| Bit 12 | **RBURST:** Burst read
| | This bit enables Burst read mode. The SDRAM controller anticipates the next read commands during the CAS latency and stores data in the Read FIFO.
| | 0: single read requests are not managed as bursts
| | 1: single read requests are always managed as bursts
| | **Note:** The corresponding bit in the FMC_SDCR2 register is read only.
| Bits 11:10 | **SDCLK[1:0]: SDRAM clock configuration**
| | These bits define the SDRAM clock period for both SDRAM banks and allow disabling the clock before changing the frequency. In this case the SDRAM must be re-initialized.
| | 00: SDCLK clock disabled
| | 01: Reserved
| | 10: SDCLK period = 2 x fmc_ker_ck periods
| | 11: SDCLK period = 3 x fmc_ker_ck periods
| | **Note:** The corresponding bits in the FMC_SDCR2 register is read only.
| Bit 9 | **WP:** Write protection
| | This bit enables Write mode access to the SDRAM bank.
| | 0: Write accesses allowed
| | 1: Write accesses ignored

| Bit 31 | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
Bits 8:7 **CAS[1:0]**: CAS Latency  
This bits sets the SDRAM CAS latency in number of memory clock cycles  
00: reserved.  
01: 1 cycle  
10: 2 cycles  
11: 3 cycles

Bit 6 **NB**: Number of internal banks  
This bit sets the number of internal banks.  
0: Two internal Banks  
1: Four internal Banks

Bits 5:4 **MWID[1:0]**: Memory data bus width.  
These bits define the memory device width.  
00: 8 bits  
01: 16 bits  
10: 32 bits  
11: reserved.

Bits 3:2 **NR[1:0]**: Number of row address bits  
These bits define the number of bits of a row address.  
00: 11 bit  
01: 12 bits  
10: 13 bits  
11: reserved.

Bits 1:0 **NC[1:0]**: Number of column address bits  
These bits define the number of bits of a column address.  
00: 8 bits  
01: 9 bits  
10: 10 bits  
11: 11 bits.

**Note:** Before modifying the RBURST or RPIPE settings or disabling the SDCLK clock, the user must first send a PALL command to make sure ongoing operations are complete.

**SDRAM Timing registers for SDRAM memory bank x (FMC_SDTRx)**

Address offset: 0x148 + 4 * (x - 1), (x = 1 to 2)  
Reset value: 0x0FFF FFFF  
This register contains the timing parameters of each SDRAM bank

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

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

Bits 27:24  **TRCD[3:0]**: Row to column delay
These bits define the delay between the Activate command and a Read/Write command in number of memory clock cycles.
- 0000: 1 cycle
- 0001: 2 cycles
- ....
- 1111: 16 cycles

Bits 23:20  **TRP[3:0]**: Row precharge delay
These bits define the delay between a Precharge command and another command in number of memory clock cycles. The TRP timing is only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRP must be programmed with the timing of the slowest device.
- 0000: 1 cycle
- 0001: 2 cycles
- ....
- 1111: 16 cycles

*Note: The corresponding bits in the FMC_SDTR2 register are don’t care.*

Bits 19:16  **TWR[3:0]**: Recovery delay
These bits define the delay between a Write and a Precharge command in number of memory clock cycles.
- 0000: 1 cycle
- 0001: 2 cycles
- ....
- 1111: 16 cycles

*Note: TWR must be programmed to match the write recovery time \((t_{WR})\) defined in the SDRAM datasheet, and to guarantee that:  
\[ TWR \geq TRAS - TRCD \text{ and } TWR \geq TRC - TRCD - TRP \]
Example: \( TRAS = 4 \) cycles, \( TRCD = 2 \) cycles. So, \( TWR \geq 2 \) cycles. \( TWR \) must be programmed to \( 0x1 \).

*If two SDRAM devices are used, the FMC_SDTR1 and FMC_SDTR2 must be programmed with the same TWR timing corresponding to the slowest SDRAM device.*
*If only one SDRAM device is used, the TWR timing must be kept at reset value (0xF) for the not used bank.*

Bits 15:12  **TRC[3:0]**: Row cycle delay
These bits define the delay between the Refresh command and the Activate command, as well as the delay between two consecutive Refresh commands. It is expressed in number of memory clock cycles. The TRC timing is only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRC must be programmed with the timings of the slowest device.
- 0000: 1 cycle
- 0001: 2 cycles
- ....
- 1111: 16 cycles

*Note: TRC must match the TRC and TRFC (Auto Refresh period) timings defined in the SDRAM device datasheet.*

*Note: The corresponding bits in the FMC_SDTR2 register are don’t care.*
Flexible memory controller (FMC)

Bits 11:8 TRAS[3:0]: Self refresh time
These bits define the minimum Self-refresh period in number of memory clock cycles.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

Bits 7:4 TXSR[3:0]: Exit Self-refresh delay
These bits define the delay from releasing the Self-refresh command to issuing the Activate command in number of memory clock cycles.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

Note: If two SDRAM devices are used, the FMC_SDTR1 and FMC_SDTR2 must be programmed with the same TXSR timing corresponding to the slowest SDRAM device.

Bits 3:0 TMRD[3:0]: Load Mode Register to Active
These bits define the delay between a Load Mode Register command and an Active or Refresh command in number of memory clock cycles.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

Note: If two SDRAM devices are connected, all the accesses performed simultaneously to both devices by the Command Mode register (Load Mode Register command) are issued using the timing parameters configured for Bank 1 (TMRD and TRAS timings) in the FMC_SDTR1 register.
The TRP and TRC timings are only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRP and TRC timings must be programmed with the timings of the slowest device.

SDRAM Command mode register (FMC_SDCMR)
Address offset: 0x150
Reset value: 0x0000 0000

This register contains the command issued when the SDRAM device is accessed. This register is used to initialize the SDRAM device, and to activate the Self-refresh and the Power-down modes. As soon as the MODE field is written, the command will be issued only to one or to both SDRAM banks according to CTB1 and CTB2 command bits. This register is the same for both SDRAM banks.

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
<th>M RD</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>rw</td>
</tr>
<tr>
<td>MRD</td>
<td>rw</td>
</tr>
<tr>
<td>NRFS</td>
<td>rw</td>
</tr>
<tr>
<td>CTB1</td>
<td>rw</td>
</tr>
<tr>
<td>CTB2</td>
<td>rw</td>
</tr>
<tr>
<td>MODE</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:23 Reserved, must be kept at reset value.
Bits 22:9 **MRD[13:0]: Mode Register definition**

This 14-bit field defines the SDRAM Mode Register content. The Mode Register is programmed using the Load Mode Register command. The MRD[13:0] bits are also used to program the extended mode register for mobile SDRAM.

Bits 8:5 **NRFS[3:0]: Number of Auto-refresh**

These bits define the number of consecutive Auto-refresh commands issued when MODE = ‘011’.

- 0000: 1 Auto-refresh cycle
- 0001: 2 Auto-refresh cycles
- ...
- 1110: 15 Auto-refresh cycles
- 1111: 16 Auto-refresh cycles

Bit 4 **CTB1: Command Target Bank 1**

This bit indicates whether the command will be issued to SDRAM Bank 1 or not.

- 0: Command not issued to SDRAM Bank 1
- 1: Command issued to SDRAM Bank 1

Bit 3 **CTB2: Command Target Bank 2**

This bit indicates whether the command will be issued to SDRAM Bank 2 or not.

- 0: Command not issued to SDRAM Bank 2
- 1: Command issued to SDRAM Bank 2

Bits 2:0 **MODE[2:0]: Command mode**

These bits define the command issued to the SDRAM device.

- 000: Normal Mode
- 001: Clock Configuration Enable
- 010: PALL (“All Bank Precharge”) command
- 011: Auto-refresh command
- 100: Load Mode Register
- 101: Self-refresh command
- 110: Power-down command
- 111: Reserved

**Note:** When a command is issued, at least one Command Target Bank bit (CTB1 or CTB2) must be set otherwise the command will be ignored.

**Note:** If two SDRAM banks are used, the Auto-refresh and PALL command must be issued simultaneously to the two devices with CTB1 and CTB2 bits set otherwise the command will be ignored.

**Note:** If only one SDRAM bank is used and a command is issued with it’s associated CTB bit set, the other CTB bit of the unused bank must be kept to 0.

**SDRAM refresh timer register (FMC_SDRTR)**

Address offset: 0x154

Reset value: 0x0000 0000

This register sets the refresh rate in number of SDCLK clock cycles between the refresh cycles by configuring the Refresh Timer Count value.

\[
\text{Refresh rate} = (\text{COUNT} + 1) \times \text{SDRAM clock frequency}
\]

\[
\text{COUNT} = (\text{SDRAM refresh period/Number of rows}) - 20
\]
Below an example of refresh rate calculation:

\[
\text{Refresh rate} = \frac{64 \text{ ms}}{8196 \text{ rows}} = 7.81 \mu s
\]

where 64 ms is the SDRAM refresh period.

\[7.81 \mu s \times 60 \text{MHz} = 468.6\]

The refresh rate must be increased by 20 SDRAM clock cycles (as in the above example) to obtain a safe margin if an internal refresh request occurs when a read request has been accepted. It corresponds to a COUNT value of ‘0000111000000’ (448).

This 13-bit field is loaded into a timer which is decremented using the SDRAM clock. This timer generates a refresh pulse when zero is reached. The COUNT value must be set at least to 41 SDRAM clock cycles.

As soon as the FMC_SDRTR register is programmed, the timer starts counting. If the value programmed in the register is ‘0’, no refresh is carried out. This register must not be reprogrammed after the initialization procedure to avoid modifying the refresh rate.

Each time a refresh pulse is generated, this 13-bit COUNT field is reloaded into the counter.

If a memory access is in progress, the Auto-refresh request is delayed. However, if the memory access and Auto-refresh requests are generated simultaneously, the Auto-refresh takes precedence. If the memory access occurs during a refresh operation, the request is buffered to be processed when the refresh is complete.

This register is common to SDRAM bank 1 and bank 2.

Note: The programmed COUNT value must not be equal to the sum of the following timings:
TWR+TRP+TRC+TRCD+4 memory clock cycles.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Re</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>Bits 15:15 Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 14 **REIE**: RES Interrupt Enable
0: Interrupt is disabled
1: An Interrupt is generated if RE = 1

Bits 13:1 **COUNT[12:0]**: Refresh Timer Count
This 13-bit field defines the refresh rate of the SDRAM device. It is expressed in number of memory clock cycles. It must be set at least to 41 SDRAM clock cycles (0x29).
Refresh rate = (COUNT + 1) x SDRAM frequency clock
COUNT = (SDRAM refresh period / Number of rows) - 20

Bit 0 **CRE**: Clear Refresh error flag
This bit is used to clear the Refresh Error Flag (RE) in the Status Register.
0: no effect
1: Refresh Error flag is cleared
SDRAM Status register (FMC_SDSR)

Address offset: 0x158
Reset value: 0x0000 0000

Bits 31:5  Reserved, must be kept at reset value.

Bits 4:3  MODES2: Status Mode for Bank 2
These bits define the Status Mode of SDRAM Bank 2.
00: Normal Mode
01: Self-refresh mode
10: Power-down mode

Bits 2:1  MODES1: Status Mode for Bank 1
These bits define the Status Mode of SDRAM Bank 1.
00: Normal Mode
01: Self-refresh mode
10: Power-down mode

Bit 0  RE: Refresh error flag
0: No refresh error has been detected
1: A refresh error has been detected
An interrupt is generated if REIE = 1 and RE = 1

23.9.6  FMC register map

The following table summarizes the FMC registers.

Table 176. FMC register map

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | FMC_BCR1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | FMC_BCR2      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | FMC_BCR3      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Reset value

0x00 FMC_BCR1 000000110 00 1010010
0x08 FMC_BCR2 000000110 00 1010010
0x10 FMC_BCR3 000000110 00 1010010

STI
### Table 176. FMC register map (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Type</th>
<th>Reset value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x18</td>
<td>FMC_BCR4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>FMC_BTR1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>FMC_BTR2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>FMC_BTR3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>FMC_BTR4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x104</td>
<td>FMC_BWTR1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10C</td>
<td>FMC_BWTR2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x114</td>
<td>FMC_BWTR3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x11C</td>
<td>FMC_BWTR4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x80</td>
<td>FMC_PCR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x64</td>
<td>FMC_SR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x88</td>
<td>FMC_FMEM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x9C</td>
<td>FMC_FMTR1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x94</td>
<td>FMC_ECCR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x140</td>
<td>FMC_SDTR1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The table above shows the register map for the Flexible Memory Controller (FMC) with various offsets, register names, and their corresponding reset values and descriptions.
Refer to Section 2.3 on page 128 for the register boundary addresses.
24 Octo-SPI interface (OCTOSPI)

24.1 Introduction

The OCTOSPI supports two frame formats used by most external serial memories such as serial PSRAMs, serial NAND and serial NOR Flash memories, HyperRAM™ and HyperFlash™ memories:

- Indirect mode: all the operations are performed using the OCTOSPI registers.
- Status polling-mode: the external memory status register is periodically read and an interrupt can be generated in case of flag setting.
- Memory-mapped mode: the external memory is memory mapped and it is seen by the system as if it was an internal memory supporting both read and write operations.

The OCTOSPI supports two frame formats:

- the classical frame format with the command, address, alternate byte, dummy cycles and data phase
- the HyperBus™ frame format

24.2 OCTOSPI main features

- Three functional modes: Indirect, Status polling, and Memory-mapped
- Read and write support in Memory-mapped mode
- Supports for single, dual, quad and octal communication
- Dual-quad mode, where 8 bits can be sent/received simultaneously by accessing two quad memories in parallel
- SDR (single-data rate) and DTR (double-transfer rate) support
- Data strobe support
- Fully programmable opcode
- Fully programmable frame format
- Support wrapped-type access to memory in read direction
- HyperBus support
- Integrated FIFO for reception and transmission
- 8, 16, and 32-bit data accesses allowed
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error.
24.3 OCTOSPI functional description

24.3.1 OCTOSPI block diagram

Figure 134. OCTOSPI block diagram for octal communication

Figure 135. OCTOSPI block diagram in Quad mode
24.3.2 OCTOSPI interface to memory modes

The OCTOSPI can operate in two different operating modes:
- Regular-command mode
- HyperBus mode

In any of these two modes, the OCTOSPI uses from 6 to 12 signals to interface with a memory, depending on the functional mode:
- nCS: chip-select used in all modes
- CLK: communication clock used in all modes
- nCLK: inverted clock used only in the 1.8 V HyperBus mode
- DQS: data strobe used mainly in Octal mode
- IO[3:0]: data bus LSB used in all modes
- IO[7:4]: data bus MSB used in Dual-quad and Octal modes

24.3.3 OCTOSPI Regular-command mode

When in Regular-command mode, the OCTOSPI communicates with the external device using commands. Each command can include five phases:
- Instruction phase
- Address phase
- Alternate-byte phase
- Dummy phase
- Data phase
Any of these phases can be configured to be skipped, but at least one of the instruction, address, alternate byte, or data phases must be present.

The nCS falls before the start of each command and rises again after each command finishes.

In Memory-mapped mode, both read and write operation are supported, as a consequence, some of the configuration registers are duplicated to specify write operations (read operations are configured using regular registers).

**Figure 137. SDR read command in Octal mode example**

The specific Regular-command mode features are configured through the registers in the 0x0100-0x01FC offset range.

### Instruction phase

During this phase, a 1- to 4-byte instruction is sent to the external device specifying the type of operation to be performed. The size of the instruction to be sent is configured in the isize[1:0] field of the OCTOSPI_CCR register and the instruction is programmed in the instruction[31:0] field of the OCTOSPI_IR register.

Most of the devices can receive instructions only 1 bit at a time from the IO0/IO signal (Single-SPI mode), the instruction phase can optionally send 2 bits at a time (over IO0/IO1 in Dual-SPI mode), 4 bits at a time (over IO0 to IO3 in Quad-SPI mode) or 8-bits at a time (over IO0 to IO7 in Octal SPI mode). This can be configured using the IMODE[2:0] field of the OCTOSPI_CCR register.

The instruction can be sent in DTR (double-transfer rate) mode on each rising and falling edge of the clock, by setting the IDTR bit in OCTOSPI_CCR.

When IMODE[2:0] = 000 in OCTOSPI_CCR, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

When in Memory-mapped mode, the instruction used for the write operation is specified in the OCTOSPI_WIR register and the instruction format is specified in the OCTOSPI_WCCR register. The instruction used for the read operation and the instruction format are specified in the regular registers OCTOSPI_IR and OCTOSPI_CCR.

### Address phase

In the address phase, 1 to 4 bytes are sent to the external device, to indicate the address of the operation. The number of address bytes to be sent is configured in the ADSIZE[1:0] field of the OCTOSPI_CCR register.

In Indirect and Automatic-polling modes, the address bytes to be sent are specified in the ADDRESS[31:0] field of the OCTOSPI_AR register. In Memory-mapped mode, the address is given directly via the AXI (from the Cortex-M core or from a DMA).
The address phase can send 1 bit at a time (over SO in Single-SPI mode), 2 bits at a time (over IO0/IO1 in Dual-SPI mode), 4 bits at a time (over IO0 to IO3 in Quad-SPI mode) or 8 bits at a time (over IO0 to IO7 in Octal SPI mode). This can be configured using the ADMODE[2:0] field of the OCTOSPI_CCR register.

The address can be sent in DTR mode (on each rising and falling edge of the clock) setting the ADDTR bit in OCTOSPI_CCR.

When ADMODE[2:0] = 000, the address phase is skipped and the command sequence proceeds directly to the next phase, if any.

In Memory-mapped mode, the address format for the write operation is specified in the OCTOSPI_WCCR register. The address format for the read operation is specified in the regular register OCTOSPI_CCR.

Alternate-bytes phase

In the alternate-bytes phase, 1 to 4 bytes are sent to the external device, generally to control the mode of operation. The number of alternate bytes to be sent is configured in the ABSIZE[1:0] field of the OCTOSPI_CCR register. The bytes to be sent are specified in the OCTOSPI_ABR register.

The alternate-bytes phase can send 1 bit at a time (over SO in Single-SPI mode), 2 bits at a time (over IO0 and IO1 in Dual-SPI mode), 4 bits at a time (over IO0 to IO3 in Quad-SPI mode) or 8 bits at a time (over IO0 to IO7 in Octal SPI mode). This can be configured using the ABMODE[2:0] field of the OCTOSPI_CCR register.

The alternate bytes can be sent in DTR mode (on each rising and falling edge of the clock) setting the ABDTR bit of OCTOSPI_CCR.

When ABMODE[2:0] = 000, the alternate-bytes phase is skipped and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when the Dual-SPI mode is used and only two cycles are used for the alternate bytes.

In this case, the firmware can use the Quad-SPI mode (ABMODE[2:0] = 011) and send a byte with bits 7 and 3 of ALTERNATE[31:0] set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low), in the OCTOSPI_IR register.

The upper two bits of the nibble to be sent are then placed in bits 4:3 of ALTERNATE[31:0] while the lower two bits are placed in bits 1 and 0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE[31:0] must be set to 0x8A (1000_1010).

In Memory-mapped mode, the alternate bytes used for the write operation are specified in the OCTOSPI_WABR register and the alternate byte format is specified in the OCTOSPI_WCCR register. The alternate bytes used for read operation and the alternate byte format are specified in the regular registers OCTOSPI_ABR and OCTOSPI_CCR.

Dummy-cycles phase

In the dummy-cycles phase, 1 to 31 cycles are given without any data being sent or received, in order to give the external device, the time to prepare for the data phase when the higher clock frequencies are used. The number of cycles given during this phase is specified in the DCYC[4:0] field of the OCTOSPI_TCR register. In both SDR and DTR modes, the duration is specified as a number of full CLK cycles.
When DCYC[4:0] = 00000, the dummy-cycles phase is skipped, and the command sequence proceeds directly to the data phase, if present.

In order to assure enough “turn-around” time for changing the data signals from the output mode to the input mode, there must be at least one dummy cycle when using the Dual-SPI, the Quad-SPI or the Octal-SPI mode, to receive data from the external device.

It is recommended to have at least five dummy cycles when using memories with DQS activated.

In Memory-mapped mode, the dummy cycles for the write operations are specified in the OCTOSPI_WTCR register. The dummy cycles for the read operation are specified in the regular register (OCTOSPI_TCR).

**Data phase**

During the data phase, any number of bytes can be sent to or received from the external device.

In Indirect mode, the number of bytes to be sent/received is specified in the OCTOSPI_DLR register. In this mode, the data to be sent to the external device must be written to the OCTOSPI_DR register, while in Indirect-read mode the data received from the external device is obtained by reading from the OCTOSPI_DR register.

In Automatic-polling mode, the number of bytes to be received is specified in the OCTOSPI_DLR register and the data received from the external device can be obtained by reading from the OCTOSPI_DR register.

In Memory-mapped mode, the data read or written, is sent or received directly over the AXI to the Cortex core or to a DMA.

The data phase can send/receive 1 bit at a time (over SO/SI in Single-SPI mode), 2 bits at a time (over IO0/IO1 in Dual-SPI mode), 4 bits at a time (over IO0 to IO3 in Quad-SPI mode) or 8 bits at a time (over IO0 to IO7 in Octal SPI mode). This can be configured using the DMODE[2:0] field of the OCTOSPI_CCR register.

The data can be sent or received in DTR mode (on each rising and falling edge of the clock) setting the DDTR bit of OCTOSPI_CCR.

When DMODE[2:0] = 000, the data phase is skipped, and the command sequence finishes immediately by raising the nCS. This configuration must be used only in Indirect-write mode.

In Memory-mapped mode, the data format for the write operation is specified in the OCTOSPI_WCCR register. The data format for the read operation is specified in the regular register OCTOSPI_CCR.

**DQS usage**

The DQS signal can be used for data strobing during the read transactions when the device toggles the DQS aligned with the data.

The DQS management can be enabled by setting the DQS enable (DQSE) bit of OCTOSPI_CCR.
24.3.4 OCTOSPI Regular-command mode signal interface

Single-SPI mode

The legacy SPI mode allows just a single bit to be sent/received serially. In this mode, the data is sent to the external device over the SO signal (whose I/O are shared with IO0). The data received from the external device arrives via SI (whose I/O are shared with IO1).

The different phases can each be configured separately to use this Single-bit mode by setting to 001 the IMODE/ADMODE/ABMODE/DMODE fields (in OCTOSPI_CCR and OCTOSPI_WCCR).

In each phase configured in Single-SPI mode:
- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0 (to deactivate the "write protect" function).
- IO3 is in output mode and forced to 1 (to deactivate the "hold" function).
- IO4/IO5/IO6/IO7 are in output mode and forced to 0.

This is the case even for the dummy phase if DMODE[2:0] = 001.

Dual-SPI mode

In Dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use Dual-SPI mode by setting to 010 the IMODE/ADMODE/ABMODE/DMODE fields (in OCTOSPI_CCR and OCTOSPI_WCCR).

In each phase configured in Dual-SPI mode:
- IO0/IO1 are at high-impedance (input) during the data phase for the read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.
- IO4/IO5/IO6/IO7 are in output mode and forced to 0.

In the dummy phase when DMODE[2:0] = 010, IO0/IO1 are always high-impedance.
Quad-SPI mode

In Quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use the Quad-SPI mode by setting to 011 the IMODE/ADMODE/ABMODE/DMODE fields (in OCTOSPI_CCR and OCTOSPI_WCCR).

In each phase configured in Quad-SPI mode:
- IO0/IO1/IO2/IO3 are all at high-impedance (input) during the data phase for the read operations, and outputs in all other cases.
- IO4/IO5/IO6/IO7 are in output mode and forced to 0.

In the dummy phase when DMODE[2:0] = 011, IO0/IO1/IO2/IO3 are all high-impedance.

IO2 and IO3 are used only in Quad-SPI mode. If none of the phases are configured to use the Quad-SPI mode, then the pins corresponding to IO2 and IO3 can be used for other functions even while the OCTOSPI is active.

Octo-SPI mode

In regular Octo-SPI mode, the eight bits are sent/received simultaneously over the IO[0:7] signals.

The different phases can each be configured separately to use the Octo-SPI mode by setting to 100 the IMODE/ADMODE/ABMODE/DMODE fields (in OCTOSPI_CCR and OCTOSPI_WCCR).

In each phase that is configured in Octal mode, IO[0:7] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE[2:0] = 100, IO[0:7] are all high-impedance.

IO[4:7] are used only in Octo-SPI mode. If none of the phases are configured to use Octo-SPI mode, then the pins corresponding to IO[4:7] can be used for other functions even while the OCTOSPI is active.

Single-data rate (SDR) mode

By default, all the phases operate in Single-data rate (SDR) mode.

In SDR mode, when the OCTOSPI is driving the IO0/SO, IO1, IO2, IO3, IO4, IO5, IO6 and IO7 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the OCTOSPI assumes that the external devices also send the data using CLK falling edge. By default (when SSHIFT = 0 in OCTOSPI_TCR), the signals are sampled using the following (rising) edge of CLK.

Figure 139. SDR write command in Octo-SPI mode example.
Double-transfer rate (DTR) mode

Each of the instruction/address/alternate/data phase can be configured to operate in Double-transfer rate (DTR) mode setting IDTR/ADDTR/ABDTR/DDTR bit in the OCTOSPI_CCR register.

In Memory-mapped mode, the DTR mode for each phases of the write operations is specified in the OCTOSPI_WCCR register. The DTR mode for each phases of the read operations is specified in the regular register (OCTOSPI_CCR).

In DTR mode, when the OCTOSPI drives the IO0/SO, IO1, IO2, IO3, IO4, IO5, IO6 and IO7 signals in the instruction/address/alternate-byte/data phases, a bit is sent or received on each of the falling and rising edges of CLK.

When receiving data in DTR mode, the OCTOSPI assumes that the external device also send the data using both CLK rising and falling edges. When DDTR = 1 in OCTOSPI_CCR, the software must clear SSHIFT in OCTOSPI_TCR. Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

In DTR mode, it is recommended to set the delay hold quarter cycle (DHQC) bit of OCTOSPI_TCR, to shift the outputs by a quarter of cycle and avoid to hold issues on the memory side.

Note: *DHQC must not be set when the prescaler value is 1, as this action leads to unpredictable behavior.*

Dual-quad mode

When the DQM = 1 in OCTOSPI_CR, the OCTOSPI is in Dual-quad mode: two external Quad-SPI devices (device A and device B) are used in order to send/receive 8 bits (or 16 bits in DTR mode) every cycle, effectively doubling the throughput as well as the capacity.

Each device (A or B) uses the same CLK and nCS signals, but each has separate IO0, IO1, IO2, and IO3 signals.

The Dual-quad mode can be used in conjunction with the Single-bit, Dual-bit, and Quad-bit modes, as well as with either the SDR or the DTR mode.

The device size, as specified in DEVSIZE[4:0] of OCTOSPI_DCR1, must reflect the total external device capacity, that is the double of the size of one individual component.

If address X is even, then the byte that the OCTOSPI gives for address X is the byte at the address X/2 of device A, and the byte that the OCTOSPI gives for address X + 1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.
When reading the status registers of the devices in Dual-quad mode, twice as many bytes must be read compared to the same read in Regular mode: if each device gives eight valid bits after the instruction for fetching the status register, then the OCTOSPI must be configured with a data length of 2 bytes (16 bits), and the OCTOSPI receives one byte from each device.

If each device gives a status of 16 bits, then the OCTOSPI must be configured to read 4 bytes to get all the status bits of both devices in Dual-quad mode. The least-significant byte of the result (in the data register) is the least-significant byte of device A status register. The next byte is the least-significant byte of device B status register. Then, the third byte of the data register is the device A second byte. The forth byte is the device B second byte (if devices have 16-bit status registers).

An even number of bytes must always be accessed in Dual-quad mode. For this reason, bit 0 of the DL[31:0] field in OCTOSPI_DLR is stuck at 1 when DQM = 1.

In Dual-quad mode, the behavior of device A interface signals is basically the same as in Normal mode. Device B interface signals have exactly the same waveforms as Device A ones during the instruction, address, alternate-byte, and dummy-cycles phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AIOx and the BIOx buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

### 24.3.5 HyperBus mode

In HyperBus mode, the OCTOSPI can communicate with the external device using the HyperBus protocol.

The HyperBus uses 11 to 12 pins depending on the operating voltage:
- IO[7:0] as bidirectional data bus
- RWDS for read and write data strobe and latency insertion (mapped on DQS pin)
- nCS
- CLK
- nCLK for 1.8 V operations

The HyperBus does not require any command specification nor any alternate bytes. As a consequence, a separate register set is used to define the timing of the transaction.

The HyperBus frame is composed of two phases:
- Command/address phase
- Data phase

The nCS falls before the start of a transaction and rises again after each transaction finishes.
The specific HyperBus features are configured through the registers in the 0x0200-0x02FC offset range.

Command/address phase

During this initial phase, the OCTOSPI sends 48 bits over IO[7:0] to specify the operations to be performed with the external device.

### Table 177. Command/address phase description

<table>
<thead>
<tr>
<th>CA bit</th>
<th>Bit name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>R/W#</td>
<td>Identifies the transaction as a read or a write</td>
</tr>
<tr>
<td>46</td>
<td>Address space</td>
<td>Indicates if the transaction accesses the memory or the register space</td>
</tr>
<tr>
<td>45</td>
<td>Burst type</td>
<td>Indicates if the burst is linear or wrapped</td>
</tr>
<tr>
<td>44-16</td>
<td>Row and upper column address</td>
<td>Selects the row and the upper column addresses</td>
</tr>
<tr>
<td>15-3</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td>2-0</td>
<td>Lower column address</td>
<td>Select the starting 16-bit word within the half page</td>
</tr>
</tbody>
</table>

The address space is configured through the memory type MTYP[2:0] field of the OCTOSPI_DCR1 register.

The total size of the device is configured in the device size DEVSIZE[4:0] field of the OCTOSPI_DCR1 register. In case of multi-chip product (MCP), the device size is the sum of all the sizes of all the dies of the MCP.

Read and write operation with initial latency

The HyperBus read and write operations need to respect two timings:
- $t_{RWR}$: minimal read/write recovery time for the device (defined in TRWR[7:0] of OCTOSPI_HLCR)
- $t_{ACC}$: access time for the device (defined in TAC[7:0] of OCTOSPI_HLCR)
During the read operation, the RWDS is used by the device, in two ways:
- during the command/address phase, to request an additional latency
- during the data phase, for data strobing

**Figure 142. HyperBus read operation with initial latency**

During the write operation the RWDS is used:
- By the device, during the command/address phase, to request an additional latency
- By the OCTOSPI, during the data phase, for write data masking

**Figure 143. HyperBus write operation with initial latency**
Read and write operation with additional latency

If the device needs an additional latency (during refresh period of a SDRAM for example), RWDS must be tied to one during one of the RWDS signals, during the command/address phase.

An additional $t_{ACC}$ duration is added by the OCTOSPI to meet the device request.

**Figure 144. HyperBus read operation with additional latency**

**Figure 145. HyperBus write operation with additional latency**

Fixed latency mode

Some devices or some applications may not want to operate with a variable latency time as described above.

The latency can be forced to $2 \times t_{ACC}$ by setting the Latency mode (LM) bit of OCTOSPI_HLCR.

In this Fixed latency mode, the state of the RWDS signal is not taken into account by the OCTOSPI and an additional latency is always added leading to a fixed $2 \times t_{ACC}$ latency time.
Write operation with no latency

Some devices can also require a zero latency for the write operations. This zero write latency can be forced by setting the write zero latency (WZL) bit of OCTOSPI_HLCR.

Latency on page-crossing during the read operations

An additional latency can be needed by some devices for the read operation when crossing pages. The initial latency must be respected for any page access, as a consequence, when the first access is close to the page boundary, a latency is automatically added at the page crossing to respect the $t_{ACC}$ time.

24.3.6 Common functionality between the Regular-command and HyperBus modes

The OCTOSPI supports some specific features common to both the Regular-command and the HyperBus modes, such as:

- Wrap support
- CS boundary and regulation

Wrap support

The OCTOSPI supports a hybrid wrap as defined by the HyperBus protocol. A hybrid wrap is also supported in the Regular-command mode.
In hybrid wrap, the transaction can continue after the initial wrap with an incremental access.

The wrap size that is supported by the target memory is configured in the wrap size (WRAPSIZE) field of the OCTOSPI device configuration register 2 (OCTOSPI_DCR2).

Wrap is supported only in memory read direction and only for data size = 4 bytes for AHB or 8 bytes for AXI. Wrapped reads are supported for both HyperBus and Regular-command modes. To enable wrapped-read access, the dedicated registers OCTOSPI_WPxxx must be programmed according to the wrapped read access characteristics. The dedicated OCTOSPI_WPxxx registers apply for both HyperBus or Regular-command mode.

If the target memory is not supporting the hybrid wrap, the WRAPSIZE field must be set to 0.

**CS boundary and refresh**

Two processes can be activated to regulate the OCTOSPI transactions:
- **CS boundary**
- **Refresh**

The CS boundary feature limits a transaction to a boundary of aligned addresses. The size of the address to be aligned with, is configured in the CS boundary CSBOUND[4:0] field of OCTOSPI_DCR3 and it is equal to $2^{\text{CSBOUND}}$.

As an example, if CSBOUND[4:0] = 0x4, the boundary is set to $2^4 = 16$ bytes. As a consequence, the nCS is released each time that the LSB address is equal to 0xF and each time that a new transaction is issued to address the next data.

If CSBOUND[4:0] = 0, the feature is disabled and a minimum value of 3 is recommended.

The CS boundary feature cannot be used for Flash memory devices in write mode since a command is necessary to program another page of the Flash memory.

The refresh feature limits the duration of the transactions to the value programmed in the REFRESH[31:0] field of OCTOSPI_DCR4. The duration is expressed in number of cycles. This allows an external RAM to perform its internal refresh operation regularly.

The refresh value must be greater than the minimal transaction size in terms of number of cycles including the command/address/alternate/dummy phases.

If CS boundary and refresh are enabled at the same time, the nCS is released on the first condition met.

**Communication regulation feature**

*Note:* This section is relevant only when communication regulation feature is supported. Refer to Section 24.3: OCTOSPI implementation.

The communication regulation feature limits the maximum length of a transaction to the value programmed in the maximum transfer MAXTRAN[7:0] field of OCTOSPI_DCR3.

If the number of clock cycles reach the MAXTRAN + 1 value, and if the second OCTOSPI requests an access, the nCS is released and a new transaction is issued to address the next data. If the second OCTOSPI does not request an access, the transaction is not stopped and the nCS is not released.

If MAXTRAN[7:0] = 0, no limitation occurs.
The MAXTRAN[7:0] value must be greater than the minimal transaction size in terms of number of cycles including the command/address/alternate/dummy phases.

The communication regulation feature cannot be used for Flash memory devices in write mode, since a command is necessary to program another page of the Flash memory. If CS boundary, refresh and communication regulation are enabled at the same time, the nCS is released on the first condition met.

24.3.7 OCTOSPI operating modes introduction

The OCTOSPI has three operating modes regardless of the low-level protocol used (either regular or HyperBus):

- Indirect mode (read or write)
- Status polling mode
- Memory-mapped mode

24.3.8 OCTOSPI Indirect mode

In Indirect mode, the commands are started by writing to the OCTOSPI registers and the data is transferred by writing or reading the data register, in a similar way to other communication peripherals.

When FMODE[1:0] = 0 in OCTOSPI_CR, the OCTOSPI is in Indirect-write mode: bytes are sent to the external device during the data phase. Data is provided by writing to OCTOSPI_DR.

When FMODE[1:0] = 01, the OCTOSPI is in Indirect-read mode: bytes are received from the external device during the data phase. Data is recovered by reading OCTOSPI_DR.

In Indirect mode, when the OCTOSPI is configured in DTR mode over eight lanes, the given starting address must be even.

The number of bytes to be read/written is specified in OCTOSPI_DLR:

- If DL[31:0] = 0xFFFF FFFF, the data length is considered undefined and the OCTOSPI simply continues to transfer data until it reaches the end of the external device (as defined by DEVSIZE). If no bytes are to be transferred, DMODE[2:0] must be set to 0 in OCTOSPI_CCR.
- If DL[31:0] = 0xFFFF FFFFF and DEVSIZE[4:0] = 0x1F (its maximum value indicating at 4-Gbyte device), the transfers continue indefinitely, stopping only after an abort request or after the OCTOSPI is disabled. After the last memory address is read (at address 0xFFFF_FFFF), reading continues with address = 0x0000_0000.

When the programmed number of bytes to be transmitted or received is reached, TCF bit is set in OCTOSPI_SR and an interrupt is generated if TCIE = 1 in OCTOSPI_CR. In the case of an undefined number of data, TCF is set when the limit of the external SPI memory is reached, according to the device size defined in OCTOSPI_DCR1.

Triggering the start of a transfer in Regular-command mode

Depending on the OCTOSPI configuration, there are three different ways to trigger the start of a transfer in Indirect mode when using Regular mode. In general, the start of transfer is triggered as soon as the software gives the last information that is necessary for the
command. More specifically in Indirect mode, a transfer starts when one of the following sequence of events occurs:

- if no address is necessary (ADMODE[2:0] = 000) and if no data needs to be provided by the software (FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to INSTRUCTION[31:0] in OCTOSPI_IR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if no data needs to be provided by the software (when FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to ADDRESS[31:0] in OCTOSPI_AR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if data needs to be provided by the software (when FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), and at the moment when a write is performed to DATA[31:0] in OCTOSPI_DR

A write to the alternate byte register OCTOSPI_ABR never triggers the communication start. If alternate bytes are required, they must have been programmed before.

As soon as a command is started, the BUSY bit is automatically set in OCTOSPI_SR.

**Triggering the start of a transfer in HyperBus mode**

Depending on the OCTOSPI configuration, there are three different ways to trigger the start of a command in Indirect mode. In general, it is triggered as soon as the firmware gives the last information that is necessary for the command, and more specifically, a communication in Indirect mode starts at the moment when:

- A write is performed to ADDRESS[31:0] (OCTOSPI_AR) in Indirect-read mode (when FMODE=01)
- A write is performed to DATA[31:0] (OCTOSPI_DR) in Indirect-write mode (when FMODE=00)

As soon as a transfer is started, the BUSY bit (OCTOSPI_SR[5]) is automatically set.

**FIFO and data management**

Data in Indirect mode passes through a 32-byte FIFO that is internal to the OCTOSPI. FLEVEL[5:0] in OCTOSPI_SR indicates how many bytes are currently being held in the FIFO.

In Indirect-write mode (FMODE[1:0] = 00), the software adds data to the FIFO when it writes in the OCTOSPI_DR. A word write adds 4 bytes to the FIFO, an half-word write adds 2 bytes, and a byte write adds only 1 byte. If the software adds too many bytes to the FIFO (more than indicated in DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

The byte/half-word accesses to the OCTOSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES[4:0] is used to define a FIFO threshold after which point the FIFO threshold flag, FTF, gets set. In Indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there is any data left in the FIFO after the last byte is read from the external device, regardless of the FTHRES[4:0] setting. In Indirect-write mode, the FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when the FTF is set. If DMAEN = 1, a DMA transfer is initiated when the FTF is set. The FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).
In Indirect-read mode, when the FIFO becomes full, the OCTOSPI temporarily stops reading bytes from the external device to avoid an overrun. Note that the reading of the external device does not restart until 4 bytes become vacant in the FIFO (when $\text{FLEVEL}[5:0] \leq 0x28$). Thus, when $\text{FTHRES}[4:0] \geq 0x29$, the application must be sure to read enough bytes to assure that the OCTOSPI starts retrieving data from the external device again. Otherwise, the FTF flag stays at 0 as long as $\text{FLEVEL}[5:0] < \text{FTHRES}[4:0]$.

The last data read in RX FIFO remains valid as long as there is no request for the next line. This means that, when the application reads several times in a row at the same location, the data is provided from the RX FIFO and not read again from the distant memory.

### 24.3.9 OCTOSPI Status-flag polling mode

In Automatic-polling mode, the OCTOSPI periodically starts a command to read a defined number of status bytes (up to 4). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value.

The access to the device begins in the same manner as in Indirect-read mode. The BUSY bit in OCTOSPI_SR goes high at this point and stays high even between the periodic accesses.

The content of MASK[31:0] in OCTOSPI_PSMAR is used to mask the data from the external device in Automatic-polling mode:

- If the MASK[n] = 0, then bit n of the result is masked and not considered.
- If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] in OCTOSPI_PSMAR, then there is a match for bit n.

If the polling match mode PMM bit in OCTOSPI_CR is 0, the AND-match mode is activated: the status match flag (SMF) is set in OCTOSPI_SR only when there is a match on all of the unmasked bits.

If PMM = 1 in OCTOSPI_CR, the OR-match mode is activated: SMF gets set if there is a match on any of the unmasked bits.

An interrupt is called when SMF = 1 if SMIE = 1.

If the Automatic-polling mode stop APMS bit is set in OCTOSPI_CR, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1 and the periodic accesses continue until there is an abort or until the OCTOSPI is disabled (EN = 0).

The OCTOSPI_DR register contains the latest received status bytes (FIFO deactivated). The content of this register is not affected by the masking used in the matching logic. The FTF status bit in OCTOSPI_SR is set as soon as a new reading of the status is complete. FTF is cleared as soon as the data is read.

In Automatic-polling mode, variable latency is not supported. As a consequence, the memory must be configured in Fixed latency.

### 24.3.10 OCTOSPI Memory-mapped mode

When configured in Memory-mapped mode, the external SPI device is seen as an internal memory.

*Note:* More than 256 Mbytes can be addressed even if the external device capacity is larger.
If an access is made to an address outside of the range defined by DEVSIZE[4:0] but still within the 256 Mbytes range, then an AXI error is given. The effect of this error depends on the AXI master that attempted the access:

- If it is the Cortex CPU, a hard-fault interrupt is generated.
- If it is a DMA, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

Byte, half-word, and word access types are all supported.

A support for execute in place (XIP) operation is implemented, where the OCTOSPI continues to load the bytes to the addresses following the most recent access. If subsequent accesses are continuous to the bytes that follow, then these operations ends up quickly since their results were pre-fetched.

By default, the OCTOSPI never stops its prefetch operation, it either keeps the previous read operation active with the nCS maintained low or it relaunches a new transfer, even if no access to the external device occurs for a long time.

Since external devices tend to consume more when the nCS is held low, the application may want to activate the timeout counter (TCEN = 1 in OCTOSPI_CR): the nCS is released after a period defined by TIMEOUT[15:0] in OCTOSPI_LPTR, when x cycles have elapsed without an access since the clock is inactive.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is an abort, or the peripheral is disabled.

24.3.11 OCTOSPI configuration introduction

The OCTOSPI configuration is done in three steps:

1. OCTOSPI system configuration
2. OCTOSPI device configuration
3. OCTOSPI mode configuration

24.3.12 OCTOSPI system configuration

The OCTOSPI is configured using the OCTOSPI_CR. The user must program:

- Functional mode with FMODE[1:0]
- Polling mode behavior if needed with PMM and APMS bits
- FIFO level with FTHRES[4:0]
- DMA usage with DMAEN
- Timeout counter usage with TCEN
- Dual-quad mode, if needed, with DQM (only for Quad-SPI configuration)

In case of an interrupt usage, the respective enable bit can also be set during this phase.

If the timeout counter is used, the timeout value is programmed in OCTOSPI_LPTR.

The DMA channel must not be enabled during the OCTOSPI configuration: it must be enabled only when the operation is fully configured, to avoid any unexpected request generation.

The DMA and OCTOSPI must be configured in a coherent manner regarding data length: the FTHRES[4:0] value in OCTOSPI must reflect the DMA burst size.
24.3.13 OCTOSPI device configuration

The parameters related to the external device targeted are configured through OCTOSPI_DCR1 and OCTOSPI_DCR2. The user must program:

- Device size with DEVSIZE[4:0]
- Chip-select minimum high time with CSHT[5:0]
- Clock mode with FRCK and CKMODE
- Device frequency with PRESCALER[7:0]

MTYP[2:0] defines the memory type to be used for 8-line modes:

- Micron mode with D0/D1 ordering in 8-data-bit mode (DMODE[2:0] = 100)
- Macronix mode with D1/D0 ordering in 8-data-bit mode (DMODE[2:0] = 100)
- HyperBus memory mode: the protocol follows the HyperBus specification, and an 8-data-bit DDR mode must be selected.
- HyperBus register mode, addressing register space: the memory-mapped accesses in this mode must be non-cacheable, or the indirect read/write modes must be used.

DEVSIZE[4:0] defines the size of external memory using the following formula:

\[
\text{Number of bytes in the device} = 2^{\text{DEVSIZE}+1}
\]

where DEVSIZE+1 is the number of address bits required to address the external device.

The external device capacity can go up to 4 Gbytes (addressed using 32 bits) in Indirect mode, but the addressable space in Memory-mapped mode is limited to 256 Mbytes.

If DQM = 1, DEVSIZE[4:0] indicates the total capacity of the two devices together.

When the OCTOSPI executes two commands, one immediately after the other, it raises the chip-select signal (nCS) high between the two commands for only one CLK cycle by default.

If the external device requires more time between commands, the chip-select high time CSHT[5:0] field can be used to specify the minimum number of CLK cycles (up to 8) for which the nCS must remain high.

**Note:** The CLK cycles are up to 64 when extended CSHT timeout feature is supported and up to 8 when it is not supported. Refer to Section 24.3: OCTOSPI implementation.

The clock mode CKMODE bit indicates the level that the CLK takes between commands (when nCS=1).

In HyperBus mode, the device timing (tACC and tRWR) and the Latency mode must be configured in OCTOSPI_HLCR.

24.3.14 OCTOSPI Regular-command mode configuration

**Indirect mode configuration**

When FMODE[1:0] = 00, the Indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the Indirect-read mode is selected and data can be read from the external device.
When the OCTOSPI is used in Indirect mode, the frames are constructed in the following way:
1. Specify a number of data bytes to read or write in OCTOSPI_DLR.
2. Specify the frame timing in OCTOSPI_TCR.
3. Specify the frame format in OCTOSPI_CCR.
4. Specify the instruction in OCTOSPI_IR.
5. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR.
6. Specify the targeted address in OCTOSPI_AR.
7. Enable the DMA channel if needed.
8. Read/write the data from/to the FIFO through OCTOSPI_DR (if no DMA usage).

If neither the address register (OCTOSPI_AR) nor the data register (OCTOSPI_DR) need to be updated for a particular command, then the command sequence starts as soon as OCTOSPI_IR is written. This is the case when both ADMODE[2:0] and DMODE[2:0] equal 000, or if just ADMODE[2:0] = 000 when in Indirect-read mode (FMODE[1:0] = 01).

When an address is required (ADMODE[2:0] ≠ 000) and the data register does not need to be written (FMODE[1:0] = 01 or DMODE[2:0] = 000), the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

In case of data transmission (FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), the communication start is triggered by a write in the FIFO through OCTOSPI_DR.

**Status-flag polling mode configuration**

The Status-flag polling mode is enabled by setting FMODE[1:0] = 10. In this mode, the programmed frame is sent and the data is retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in OCTOSPI_DLR, it is ignored and only 4 bytes are read. The periodicity is specified in OCTOSPI_PIR.

Once the status data has been retrieved, the following can be processed:
- Set SMF (an interrupt is generated if enabled).
- Stop automatically the periodic retrieving of the status bytes.

The received value can be masked with the value stored in OCTOSPI_PSMKR, and can be ORed or ANDed with the value stored in OCTOSPI_PSMAR.

In case of a match, SMF is set and an interrupt is generated if enabled; The OCTOSPI can be automatically stopped if AMPS is set. In any case, the latest retrieved value is available in OCTOSPI_DR.
When the OCTOSPI is used in Auto-polling mode, the frames are constructed in the following way:
1. Specify the input mask in OCTOSPI_PSMKR.
2. Specify the comparison value in OCTOSPI_PSMAR.
3. Specify the read period in OCTOSPI_PIR.
4. Specify a number of data bytes to read in OCTOSPI_DLR.
5. Specify the frame timing in OCTOSPI_TCR.
6. Specify the frame format in OCTOSPI_CCR.
7. Specify the instruction in OCTOSPI_IR.
8. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR.
9. Specify the optional targeted address in OCTOSPI_AR.

If the address register (OCTOSPI_AR) does not need to be updated for a particular command, then the command sequence starts as soon as OCTOSPI_CCR is written. This is the case when ADMODE[2:0] = 000.

When an address is required (ADMODE[2:0] ≠ 000), the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

Memory-mapped mode configuration

In Memory-mapped mode, the external device is seen as an internal memory but with some latency during accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the Flash memory using memory-mapped writes, as the internal flags for erase or programming status have to be polled.

Memory-mapped mode is entered by setting FMODE[1:0] = 11 in OCTOSPI_CR.

The programmed instruction and frame are sent when an AXI master is accessing the memory mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to OCTOSPI_DR in this mode returns zero.

The data length register (OCTOSPI_DLR) has no meaning in Memory-mapped mode.

When the OCTOSPI is used in Memory-mapped mode, the frames are constructed in the following way:
1. Specify the frame timing in OCTOSPI_TCR for read operation.
2. Specify the frame format in OCTOSPI_CCR for read operation.
3. Specify the instruction in OCTOSPI_IR.
4. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR for read operation.
5. Specify the frame timing in OCTOSPI_WTCR for write operation.
6. Specify the frame format in OCTOSPI_WCCR for write operation.
7. Specify the instruction in OCTOSPI_WIR.
8. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_WABR for read operation.
All the configuration operations must be completed before the first access to the memory area. On the first access, the OCTOSPI becomes busy, and no further configuration is allowed.

**OCTOSPI delayed data sampling when no DQS is used**

By default, when no DQS is used, the OCTOSPI samples the data driven by the external device one half of a CLK cycle after the external device drives the signal.

In case of any external signal delays, it may be useful to sample the data later. Using the SSHIFT bit in OCTOSPI_TCR, the sampling of the data can be shifted by half of a CLK cycle.

The firmware must clear SSHIFT when the data phase is configured in DTR mode (DDTR = 1).

**OCTOSPI delayed data sampling when DQS is used**

When external DQS is used as a sampling clock, it can be shifted in time to compensate the data propagation delay. This shift is performed by an external delay block located outside the OCTOSPI. The control of this feature depends on the device implementation (see the product reference manual for more details).

In certain configuration cases, this external delay block is implemented but is not useful, so it can be bypassed by setting DLYBYP bit in OCTOSPI_DCR1.

**Sending the instruction only once (SIOO)**

A Flash memory can provide a mode where an instruction must be sent only with the first command sequence, while subsequent commands start directly with the address. The user can take advantage of this type of features using the SIOO bit in OCTOSPI_CCR.

The SIOO is valid for Memory-mapped mode only. If this bit is set, the instruction is sent only for the first command following a write to OCTOSPI_CCR.

Subsequent command sequences skip the instruction phase, until there is a write to OCTOSPI_CCR. The SIOO has no effect when IMODE[1:0] = 00 (no instruction).

SIOO mode is not supported when any of the communication regulation, CS boundary or refresh features are used.

24.3.15 **OCTOSPI HyperBus mode configuration**

**Indirect mode configuration**

When FMODE[1:0] = 00, the Indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the Indirect-read mode is selected where data can be read from the external device.

When the OCTOSPI is used in Indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in OCTOSPI_DL R.
2. Specify the targeted address in OCTOSPI_AR.
3. Enable the DMA channel if needed.
4. Read/write the data from/to the FIFO through OCTOSPI_DR (if no DMA usage).
In Indirect-read mode, the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

In Indirect-write mode, the communication start is triggered by a write in the FIFO through OCTOSPI_DR.

**Status-flag polling mode configuration**

The Status-flag polling mode is enabled setting FMODE[1:0] = 10. In this mode, the programmed frame is sent and the data is retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in OCTOSPI_DLR, it is ignored and only 4 bytes are read. The periodicity is specified in OCTOSPI_PIR.

Once the status data has been retrieved, it can be internally processed to:
- Set SMF (an interrupt is generated if enabled).
- Stop automatically the periodic retrieving of the status bytes.

The received value can be masked with the value stored in OCTOSPI_PSMKR and can be ORed or ANDed with the value stored in OCTOSPI_PSMAR.

In case of a match, SMF is set and an interrupt is generated if enabled. The OCTOSPI can be automatically stopped if AMPS is set.

In any case, the latest retrieved value is available in OCTOSPI_DR.

When the OCTOSPI is used in Auto-polling mode, the frames are constructed in the following way:

1. Specify the input mask in OCTOSPI_PSMKR.
2. Specify the comparison value in OCTOSPI_PSMAR.
3. Specify the read period in OCTOSPI_PIR.
4. Specify a number of data bytes to read in OCTOSPI_DLR.
5. Specify the targeted address in OCTOSPI_AR.

The command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

**Memory-mapped mode configuration**

In Memory-mapped mode, the external device is seen as an internal memory but with some latency during the accesses. Read and write operations are allowed to the external device in this mode.

The Memory-mapped mode is entered by setting FMODE[1:0] = 11. The programmed instruction and frame is sent when an AXI master is accessing the memory mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to OCTOSPI_DR in this mode returns zero.

The data length register (OCTOSPI_DLR) has no meaning in Memory-mapped mode.

All the configuration operation must be completed prior to the first access to the memory area. On the first access, the OCTOSPI becomes busy, and no configuration is allowed.
24.3.16 OCTOSPI error management

A error can be generated in the following cases:

- In Indirect mode or Status-flag polling mode, when a wrong address has been programmed in OCTOSPI_AR (according to the device size defined by DEVSIZE[4:0]). This sets TEF and an interrupt is generated if enabled.
- In Indirect mode, if the address plus the data length exceed the device size. TEF is set as soon as the access is triggered.
- In Memory-mapped mode when an out-of-range access is done by an AXI master, it generates an AXI error as a response to the faulty AXI request.
- An access to the memory-mapped area when the Memory-mapped mode is disabled, generates an AXI error as a response to the faulty AXI request.

The OCTOSPI generates an AXI slave error in the following situations:

- Memory mapped mode is disabled and an AXI read request occurs.
- Read or write address exceeds the size of the external memory.
- Abort is received while a read or write burst is ongoing.
- OCTOSPI is disabled while a read or write burst is ongoing.
- Write wrap burst received.
- Write request received while the DQSE bit in the WCCR register is 0, which means that the DQS output is disabled.
- Write request received while DMODE[2:0] = 000 (no data phase), except when MTYPE[2:0] is HyperBus.
- Illegal access size when wrap read burst. For AHB this means the HSIZE is different from 4 bytes (only for Memory-mapped mode) and for AXI this means the ARSIZE is different from 8 bytes (only for Memory-mapped mode).
- Illegal wrap size when receiving read wrap burst with size different from 4 bytes for AHB or 8 bytes for AXI (only for Memory-mapped mode).

24.3.17 OCTOSPI busy bit and abort functionality

Once the OCTOSPI starts an operation with the external device, BUSY is automatically set in OCTOSPI_SR.

In Indirect mode, BUSY is reset once the OCTOSPI has completed the requested command sequence and the FIFO is empty.

In Automatic-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1 or due to an abort.

After the first access in Memory-mapped mode, BUSY goes low only on an abort.

Any operation can be aborted by setting the ABORT bit in OCTOSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

Before setting ABORT, the software must ensure that all the current transactions are finished using the synchronization barriers.

Note: Some devices may misbehave if a write operation to a status register is aborted.
24.3.18 OCTOSPI reconfiguration or deactivation

Prior to any OCTOSPI reconfiguration, the software must ensure that all the transactions are completed:

- After a Memory-mapped write, the software must perform a dummy read followed by a synchronization barrier, then an abort.
- After a Memory-mapped read, the software must perform a synchronization barrier then an abort.

24.3.19 nCS behavior

By default, the nCS is high, deselecting the external device. The nCS falls before an operation begins and rises as soon as it finishes.

When CKMODE = 0 (Mode 0: CLK stays low when no operation is in progress), the nCS falls one CLK cycle before an operation first rising CLK edge, and the nCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

![Figure 148. nCS when CKMODE=0 (T = CLK period)](image)

When CKMODE = 1 (Mode 3: CLK goes high when no operation is in progress) and when in SDR mode, nCS falls one CLK cycle before an operation first rising CLK edge, and the nCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

![Figure 149. CS when CKMODE=1 in SDR mode (T = CLK period)](image)

When the CKMODE = 1 (Mode 3) and DDTR = 1 (data DTR mode), the nCS falls one CLK cycle before an operation first rising CLK edge, and the nCS rises one CLK cycle after the operation final active rising CLK edge (see the figure below). Because the DDR operations...
must finish with a falling edge, the CLK is low when the nCS rises, and the CLK rises back up one half of a CLK cycle afterwards.

**Figure 150. nCS when CKMODE=1 in DDTR mode (T = CLK period)**

When the FIFO stays full during a read operation, or if the FIFO stays empty during a write operation, the operation stalls and the CLK stays low until the software services the FIFO. If an abort occurs when an operation is stalled, the nCS rises just after the abort is requested and then the CLK rises one half of a CLK cycle later (see the figure below).

**Figure 151. nCS when CKMODE=1 with an abort (T = CLK period)**

When not in Dual-quad mode (DQM = 0), only device A is accessed and thus the BnCS stays high. In Dual-quad mode, the BnCS behaves exactly the same as the AnCS. Thus, if there is a device B and if the application always stays in Dual-quad mode, then the device B may use the AnCS and the pin outputting BnCS can be used for other functions.

### 24.4 OCTOSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available to provide more flexibility.
### OCTOSPI registers

#### OCTOSPI control register (OCTOSPI_CR)

Address offset: 0x0000  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>FMODE[1:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>PMM</td>
<td>APMS</td>
<td>TOIE</td>
<td>SMIE</td>
<td>FTIE</td>
<td>TCIE</td>
<td>TEIE</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

- **Bits 31:30** Reserved, must be kept at reset value.
- **Bits 29:28** **FMODE[1:0]:** Functional mode  
  This field defines the OCTOSPI functional mode of operation.  
  00: Indirect-write mode  
  01: Indirect-read mode  
  10: Automatic-polling mode  
  11: Memory-mapped mode  
  If DMAEN = 1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE[1:0] value.  
  This field can be written only when BUSY = 0. In such case, the request from OCTOSPI to DMA becomes inactive.
- **Bits 27:24** Reserved, must be kept at reset value.
- **Bit 23** **PMM:** Polling match mode  
  This bit indicates which method must be used to determine a match during the Automatic-polling mode.  
  0: AND-match mode, SMF is set if all the unmasked bits received from the device match the corresponding bits in the match register.  
  1: OR-match mode, SMF is set if any of the unmasked bits received from the device matches its corresponding bit in the match register.  
  This bit can be modified only when BUSY = 0.
Bit 22  **APMS**: Automatic-poll mode stop  
This bit determines if the automatic polling is stopped after a match.  
0: Automatic-polling mode is stopped only by abort or by disabling the OCTOSPI.  
1: Automatic-polling mode stops as soon as there is a match.  
This bit can be modified only when BUSY=0

Bit 21  Reserved, must be kept at reset value.

Bit 20  **TOIE**: Timeout interrupt enable  
This bit enables the timeout interrupt.  
0: Interrupt disabled  
1: Interrupt enabled

Bit 19  **SMIE**: Status match interrupt enable  
This bit enables the status match interrupt.  
0: Interrupt disabled  
1: Interrupt enabled

Bit 18  **FTIE**: FIFO threshold interrupt enable  
This bit enables the FIFO threshold interrupt.  
0: Interrupt disabled  
1: Interrupt enabled

Bit 17  **TCIE**: Transfer complete interrupt enable  
This bit enables the transfer complete interrupt.  
0: Interrupt disabled  
1: Interrupt enabled

Bit 16  **TEIE**: Transfer error interrupt enable  
This bit enables the transfer error interrupt.  
0: Interrupt disabled  
1: Interrupt enabled

Bits 15:13  Reserved, must be kept at reset value.

Bits 12:8  **FTHRES[4:0]**: FIFO threshold level  
This field defines, in Indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag FTF in OCTOSPI_SR, to be set.  
0: FTF is set if there are 1 or more free bytes available to be written to in the FIFO in Indirect-write mode, or if there are 1 or more valid bytes can be read from the FIFO in Indirect-read mode.  
1: FTF is set if there are 2 or more free bytes available to be written to in the FIFO in Indirect-write mode, or if there are 2 or more valid bytes can be read from the FIFO in Indirect-read mode.  
...  
31: FTF is set if there are 32 free bytes available to be written to in the FIFO in Indirect-write mode, or if there are 32 valid bytes can be read from the FIFO in Indirect-read mode.

If DMAEN = 1, the DMA controller for the corresponding channel must be disabled before changing the FTHRES[4:0] value.
Bit 7 **FSEL**: Flash select
This bit selects the Flash memory to be addressed in Single/Dual/Quad mode in Single-Flash mode (when DQM = 0).
- 0: FLASH 1 selected (data exchanged over IO[3:0])
- 1: FLASH 2 selected (data exchanged over IO[7:4])
This bit is ignored when DQM = 1 or when Octal mode is selected.
This bit can be modified only when BUSY = 0.

Bit 6 **DQM**: Dual-quad mode
This bit activates the Dual-quad mode, where two external devices are used simultaneously to double the throughput and the capacity
- 0: Dual-quad mode disabled
- 1: Dual-quad mode enabled
This bit can be modified only when BUSY = 0.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **TCEN**: Timeout counter enable
This bit is valid only when the Memory-mapped mode (FMODE[1:0] = 11) is selected. This bit enables the timeout counter.
- 0: Timeout counter is disabled, and thus the chip-select (nCS) remains active indefinitely after an access in Memory-mapped mode.
- 1: Timeout counter is enabled, and thus the chip-select is released in the Memory-mapped mode after TIMEOUT[15:0] cycles of external device inactivity.
This bit can be modified only when BUSY = 0.

Bit 2 **DMAEN**: DMA enable
In Indirect mode, the DMA can be used to input or output data via OCTOSPI_DR. DMA transfers are initiated when FTF is set.
- 0: DMA disabled for Indirect mode
- 1: DMA enabled for Indirect mode

*Note: Resetting the DMAEN bit while a DMA transfer is ongoing, breaks the handshake with the DMA. Do not write this bit during DMA operation.*

Bit 1 **ABORT**: Abort request
This bit aborts the on-going command sequence. It is automatically reset once the abort is completed. This bit stops the current transfer.
- 0: No abort requested
- 1: Abort requested

*Note: This bit is always read as 0.*

Bit 0 **EN**: Enable
This bit enables the OCTOSPI.
- 0: OCTOSPI disabled
- 1: OCTOSPI enabled

*Note: The DMA request may be aborted without having received the ACK. In this case, the EN bit is cleared during the operation.*
### OCTOSPI Device Configuration Register 1 (OCTOSPI_DCR1)

Address offset: 0x0008  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:27</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 26:24</td>
<td><strong>MTYP[2:0]</strong>: Memory type</td>
</tr>
<tr>
<td></td>
<td>This bit indicates the type of memory to be supported.</td>
</tr>
<tr>
<td></td>
<td>000: Micron mode, D0/D1 ordering in DTR 8-data-bit mode. Regular SPI protocol in Octal/Quad/Dual/Single modes</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> In this mode, DQS signal polarity is inverted with respect to the memory clock signal. This is the default value and care must be taken to change MTYP[2:0] for memories different from Micron.</td>
</tr>
<tr>
<td></td>
<td>001: Macronix mode, D1/D0 ordering in DTR 8-data-bit mode. Regular SPI protocol in Octal/Quad/Dual/Single modes</td>
</tr>
<tr>
<td></td>
<td>010: Standard mode</td>
</tr>
<tr>
<td></td>
<td>011: Macronix RAM mode, D1/D0 ordering in DTR 8-data-bit mode. Regular SPI protocol in Octal/Quad/Dual/Single modes with dedicated address mapping.</td>
</tr>
<tr>
<td></td>
<td>100: HyperBus memory mode, the protocol follows the HyperBus specification. 8-data-bit DTR mode must be selected.</td>
</tr>
<tr>
<td></td>
<td>101: HyperBus register mode, addressing register space. The memory-mapped accesses in this mode must be non-cacheable, or Indirect read/write modes must be used.</td>
</tr>
<tr>
<td></td>
<td>110-111: Reserved</td>
</tr>
<tr>
<td>Bit 23:21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 20:16</td>
<td><strong>DEVSIZE[4:0]</strong>: Device size</td>
</tr>
<tr>
<td></td>
<td>This field defines the size of the external device using the following formula:</td>
</tr>
<tr>
<td></td>
<td>Number of bytes in device = 2^{DEVSIZE+1}.</td>
</tr>
<tr>
<td></td>
<td>DEVSIZE+1 is effectively the number of address bits required to address the external device. The device capacity can be up to 4 Gbytes (addressed using 32-bits) in Indirect mode, but the addressable space in Memory-mapped mode is limited to 256 Mbytes.</td>
</tr>
<tr>
<td></td>
<td>In Regular-command mode, if DQM = 1, DEVSIZE[4:0] indicates the total capacity of the two devices together.</td>
</tr>
<tr>
<td>Bit 15:11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
24.5.3 OCTOSPI device configuration register 2 (OCTOSPI_DCR2)

Address offset: 0x000C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>DLYBYP: Delay block bypass</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The internal sampling clock (called feedback clock) or the DQS data strobe external signal is delayed by the delay block (for more details on this block, refer to the dedicated section of the reference manual as it is not part of the OCTOSPI peripheral).</td>
</tr>
<tr>
<td>1</td>
<td>The delay block is bypassed, so the internal sampling clock or the DQS data strobe external signal is not affected by the delay block. The delay is shorter than when the delay block is not bypassed, even with the delay value set to minimum value in delay block.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>FRCK: Free running clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CLK is not free running.</td>
</tr>
<tr>
<td>1</td>
<td>CLK is free running (always provided).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>CKMODE: Mode 0/Mode 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CLK must stay low while nCS is high (chip-select released). This is referred to as Mode 0.</td>
</tr>
<tr>
<td>1</td>
<td>CLK must stay high while nCS is high (chip-select released). This is referred to as Mode 3.</td>
</tr>
</tbody>
</table>

Note: 7: nCS stays high for at least 8 cycles between external device commands. When the extended CSHT timeout feature is not supported, CSHT[5:3] are reserved and the number of cycles is limited to 8. Refer to Section 24.3: OCTOSPI implementation.
Bits 31:19  Reserved, must be kept at reset value.

Bits 18:16  \textbf{WRAPSIZE}[2:0]: Wrap size

This field indicates the wrap size to which the memory is configured. For memories which have a separate command for wrapped instructions, this field indicates the wrap-size associated with the command held in the OCTOSPI1\_WPIR register.

- 000: Wrapped reads are not supported by the memory.
- 001: Reserved
- 010: External memory supports wrap size of 16 bytes.
- 011: External memory supports wrap size of 32 bytes.
- 100: External memory supports wrap size of 64 bytes.
- 101: External memory supports wrap size of 128 bytes.
- 110-111: Reserved

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:0  \textbf{PRESCALER}[7:0]: Clock prescaler

This field defines the scaler factor for generating the CLK based on the kernel clock (value + 1).

- 0: F\textsubscript{CLK} = F\textsubscript{KERNEL}, kernel clock used directly as OCTOSPI CLK (prescaler bypassed). In this case, if the DDR mode is used, it is mandatory to provide to the OCTOSPI a kernel clock that has 50\% duty-cycle.
- 255: F\textsubscript{CLK} = F\textsubscript{KERNEL}/256

For odd clock division factors, the CLK duty cycle is not 50\%. The clock signal remains high one cycle longer than it stays low.

This field can be modified only when BUSY = 0.

### 24.5.4  OCTOSPI device configuration register 3 (OCTOSPI\_DCR3)

Address offset: 0x0010

Reset value: 0x0000 0000

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

Bits 31:21  Reserved, must be kept at reset value.
24.5.5 OCTOSPI device configuration register 4 (OCTOSPI_DCR4)

Address offset: 0x0014
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>REFRESH[31:16]: Refresh rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This field enables the refresh rate feature.</td>
</tr>
<tr>
<td></td>
<td>The nCS is released every REFRESH+1 clock cycles for writes, and REFRESH+4 clock cycles for reads.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> These two values can be extended with few clock cycles when refresh occurs during a byte transmission in single, dual or quad mode, because the byte transmission must be completed.</td>
</tr>
<tr>
<td></td>
<td>0: Refresh disabled</td>
</tr>
<tr>
<td></td>
<td>others: Maximum communication length is set to REFRESH+1 clock cycles.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>REFRESH[15:0]:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Refresh rate</td>
</tr>
<tr>
<td></td>
<td>This field enables the refresh rate feature.</td>
</tr>
<tr>
<td></td>
<td>The nCS is released every REFRESH+1 clock cycles for writes, and REFRESH+4 clock cycles for reads.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> These two values can be extended with few clock cycles when refresh occurs during a byte transmission in single, dual or quad mode, because the byte transmission must be completed.</td>
</tr>
<tr>
<td></td>
<td>0: Refresh disabled</td>
</tr>
<tr>
<td></td>
<td>others: Maximum communication length is set to REFRESH+1 clock cycles.</td>
</tr>
</tbody>
</table>
### OCTOSPI status register (OCTOSPI_SR)

Address offset: 0x0020  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 15:8</td>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:14  Reserved, must be kept at reset value.

Bits 13:8  **FLEVEL[5:0]:** FIFO level  
This field gives the number of valid bytes that are being held in the FIFO. FLEVEL=0 when the FIFO is empty, and 32 when it is full.  
In Automatic-status polling mode, FLEVEL is zero.

Bits 7:6  Reserved, must be kept at reset value.

Bit 5  **BUSY:** Busy  
This bit is set when an operation is ongoing. It is cleared automatically when the operation with the external device is finished and the FIFO is empty.

Bit 4  **TOF:** Timeout flag  
This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3  **SMF:** Status match flag  
This bit is set in Automatic-polling mode when the unmasked received data matches the corresponding bits in the match register (OCTOSPI_PSMAR).  
It is cleared by writing 1 to CSMF.

Bit 2  **FTF:** FIFO threshold flag  
In Indirect mode, this bit is set when the FIFO threshold has been reached, or if there is any data left in the FIFO after the reads from the external device are complete.  
It is cleared automatically as soon as the threshold condition is no longer true.  
In Automatic-polling mode this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1  **TCF:** Transfer complete flag  
This bit is set in Indirect mode when the programmed number of data has been transferred or in any mode when the transfer has been aborted. It is cleared by writing 1 to CTCF.

Bit 0  **TEF:** Transfer error flag  
This bit is set in Indirect mode when an invalid address is being accessed in Indirect mode.  
It is cleared by writing 1 to CTEF.
24.5.7 OCTOSPI flag clear register (OCTOSPI_FCR)

Address offset: 0x0024
Reset value: 0x0000 0000

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

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 CTOF: Clear timeout flag
Writing 1 clears the TOF flag in the OCTOSPI_SR register.

Bit 3 CSMF: Clear status match flag
Writing 1 clears the SMF flag in the OCTOSPI_SR register.

Bit 2 Reserved, must be kept at reset value.

Bit 1 CTCF: Clear transfer complete flag
Writing 1 clears the TCF flag in the OCTOSPI_SR register.

Bit 0 CTEF: Clear transfer error flag
Writing 1 clears the TEF flag in the OCTOSPI_SR register.

24.5.8 OCTOSPI data length register (OCTOSPI_DLR)

Address offset: 0x0040
Reset value: 0x0000 0000

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

| DL[15:0] | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |


24.5.9 OCTOSPI address register (OCTOSPI_AR)

Address offset: 0x0048
Reset value: 0x0000 0000

Bits 31:0  ADDRESS[31:0]: Address

Address to be sent to the external device. In HyperFlash mode, this field must be even as
this protocol is 16-bit word oriented. Writes to this field are ignored when BUSY = 1 or when
FMODE = 11 (Memory-mapped mode).

24.5.10 OCTOSPI data register (OCTOSPI_DR)

Address offset: 0x0050
Reset value: 0x0000 0000

Bits 31:0  DL[31:0]: Data length

Number of data to be retrieved (value+1) in Indirect and Status-polling modes. A value not
greater than 3 (indicating 4 bytes) must be used for status polling-mode.

All 1s in Indirect mode means undefined length, where OCTOSPI continues until the end of
the memory, as defined by DEVSIZE.
0x0000_0000: 1 byte is to be transferred.
0x0000_0001: 2 bytes are to be transferred.
0x0000_0002: 3 bytes are to be transferred.
0x0000_0003: 4 bytes are to be transferred.
...
0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred.
0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred.
0xFFFF_FFFF: undefined length; all bytes, until the end of the external device, (as defined
by DEVSIZE) are to be transferred. Continue reading indefinitely if DEVSIZE = 0x1F.
DL[0] is stuck at 1 in Dual-quad mode (DQM = 1) even when 0 is written to this bit, thus
assuring that each access transfers an even number of bytes.
This field has no effect when in Memory-mapped mode (FMODE = 10).
This field can be written only when BUSY = 0.
Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device

In Indirect-write mode, data written to this register is stored on the FIFO before it is sent to the external device during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In Indirect-read mode, reading this register gives (via the FIFO) the data that was received from the external device. If the FIFO does not have as many bytes as requested by the read operation and if BUSY = 1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In Automatic-polling mode, this register contains the last data read from the external device (without masking).

Word, half-word, and byte accesses to this register are supported. In Indirect-write mode, a byte write adds 1 byte to the FIFO, a half-word write 2 bytes, and a word write 4 bytes. Similarly, in Indirect-read mode, a byte read removes 1 byte from the FIFO, a halfword read 2 bytes, and a word read 4 bytes. Accesses in Indirect mode must be aligned to the bottom of this register: A byte read must read DATA[7:0] and a half-word read must read DATA[15:0].

### 24.5.11 OCTOSPI polling status mask register (OCTOSPI _PSMKR)

Address offset: 0x0080

Reset value: 0x0000 0000

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

**MASK[31:16]**

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

**MASK[15:0]**

Bits 31:0 **MASK[31:0]**: Status mask

Mask to be applied to the status bytes received in Polling mode

For bit n:

0: Bit n of the data received in Automatic-polling mode is masked and its value is not considered in the matching logic.

1: Bit n of the data received in Automatic-polling mode is unmasked and its value is considered in the matching logic.

This field can be written only when BUSY = 0.
### 24.5.12 OCTOSPI polling status match register (OCTOSPI_PSMAR)

- **Address offset:** 0x0088
- **Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>MATCH[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

- **Bits 31:0 MATCH[31:0]:** Status match
  - Value to be compared with the masked status register to get a match
  - This field can be written only when BUSY = 0.

### 24.5.13 OCTOSPI polling interval register (OCTOSPI_PIR)

- **Address offset:** 0x0090
- **Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>INTERVAL[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

- **Bits 31:16 Reserved, must be kept at reset value.**
- **Bits 15:0 INTERVAL[15:0]:** Polling interval
  - Number of CLK cycle between a read during the automatic polling phases
  - This field can be written only when BUSY = 0.

### 24.5.14 OCTOSPI communication configuration register (OCTOSPI_CCR)

- **Address offset:** 0x0100
- **Reset value:** 0x0000 0000

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bit 31  **SIOO**: Send instruction only once mode
See *Sending the instruction only once (SIOO)*. This bit has no effect when IMODE = 00.
- 0: Send instruction on every transaction
- 1: Send instruction only for the first command
This field can be written only when BUSY = 0.

Bit 30  Reserved, must be kept at reset value.

Bit 29  **DQSE**: DQS enable
- This bit enables the data strobe management.
  - 0: DQS disabled
  - 1: DQS enabled

Bit 28  Reserved, must be kept at reset value.

Bit 27  **DDTR**: Data double transfer rate
- This bit sets the DTR mode for the data phase.
  - 0: DTR mode disabled for data phase
  - 1: DTR mode enabled for data phase
This field can be written only when BUSY = 0.

Bits 26:24  **DMODE[2:0]**: Data mode
- This field defines the data phase's mode of operation.
  - 000: No data
  - 001: Data on a single line
  - 010: Data on two lines
  - 011: Data on four lines
  - 100: Data on eight lines
  - 101-111: Reserved
This field can be written only when BUSY = 0.

Bits 23:22  Reserved, must be kept at reset value.

Bits 21:20  **ABSIZE[1:0]**: Alternate bytes size
- This bit defines alternate bytes size.
  - 00: 8-bit alternate bytes
  - 01: 16-bit alternate bytes
  - 10: 24-bit alternate bytes
  - 11: 32-bit alternate bytes
This field can be written only when BUSY = 0.

Bit 19  **ABDTR**: Alternate bytes double transfer rate
- This bit sets the DTR mode for the alternate bytes phase.
  - 0: DTR mode disabled for alternate bytes phase
  - 1: DTR mode enabled for alternate bytes phase
This field can be written only when BUSY = 0.

Bits 18:16  **ABMODE[2:0]**: Alternate-byte mode
- This field defines the Alternate byte phase's mode of operation.
  - 000: No alternate bytes
  - 001: Alternate bytes on a single line
  - 010: Alternate bytes on two lines
  - 011: Alternate bytes on four lines
  - 100: Alternate bytes on eight lines
  - 101-111: Reserved
This field can be written only when BUSY = 0.
Bits 15:14  Reserved, must be kept at reset value.

Bits 13:12  **ADSIZE[1:0]**: Address size
   This field defines address size.
   00: 8-bit address
   01: 16-bit address
   10: 24-bit address
   11: 32-bit address
   This field can be written only when BUSY = 0.

Bit 11  **ADDTR**: Address double transfer rate
   This bit sets the DTR mode for the address phase.
   0: DTR mode disabled for address phase
   1: DTR mode enabled for address phase
   This field can be written only when BUSY = 0.

Bits 10:8  **ADMODE[2:0]**: Address mode
   This field defines the Address phase’s mode of operation.
   000: No address
   001: Address on a single line
   010: Address on two lines
   011: Address on four lines
   100: Address on eight lines
   101-111: Reserved
   This field can be written only when BUSY = 0.

Bits 7:6  Reserved, must be kept at reset value.

Bits 5:4  **ISIZE[1:0]**: Instruction size
   This bit defines instruction size.
   00: 8-bit instruction
   01: 16-bit instruction
   10: 24-bit instruction
   11: 32-bit instruction
   This field can be written only when BUSY = 0.

Bit 3  **IDTR**: Instruction double transfer rate
   This bit sets the DTR mode for the instruction phase.
   0: DTR mode disabled for instruction phase
   1: DTR mode enabled for instruction phase
   This field can be written only when BUSY = 0.

Bits 2:0  **IMODE[2:0]**: Instruction mode
   This field defines the Instruction phase’s mode of operation.
   000: No instruction
   001: Instruction on a single line
   010: Instruction on two lines
   011: Instruction on four lines
   100: Instruction on eight lines
   101-111: Reserved
   This field can be written only when BUSY = 0.
# Octo-SPI interface (OCTOSPI)

## 24.5.15 OCTOSPI timing configuration register (OCTOSPI_TCR)

Address offset: 0x0108  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27:5</th>
<th>Bit 4:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>SSHIFT</td>
<td>Res.</td>
<td>DHQC</td>
<td>Res.</td>
<td>DCYC[4:0]</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>Res.</td>
<td>rw</td>
<td>Res.</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bit 31** Reserved, must be kept at reset value.

**Bit 30** **SSSHIFT**: Sample shift  
By default, the OCTOSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.  
This bit allows the data to be sampled later in order to consider the external signal delays.  
0: No shift  
1: 1/2 cycle shift  
The software must ensure that SSHIFT = 0 when the data phase is configured in DTR mode (when DDTR = 1.)  
This field can be modified only when BUSY = 0.

**Bit 29** Reserved, must be kept at reset value.

**Bit 28** **DHQC**: Delay hold quarter cycle  
0: No delay hold  
1: 1/4 cycle hold

**Bits 27:5** Reserved, must be kept at reset value.

**Bits 4:0** **DCYC[4:0]**: Number of dummy cycles  
This field defines the duration of the dummy phase.  
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).  
It is recommended to have at least six dummy cycles when using memories with DQS activated.  
This field can be written only when BUSY = 0.

## 24.5.16 OCTOSPI instruction register (OCTOSPI_IR)

Address offset: 0x0110  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27:5</th>
<th>Bit 4:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**INSTRUCTION[31:16]**

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27:5</th>
<th>Bit 4:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**INSTRUCTION[15:0]**
24.5.17 OCTOSPI alternate bytes register (OCTOSPI_ABR)

Address offset: 0x0120
Reset value: 0x0000 0000

Bits 31:0 \textbf{INSTRUCTION}[31:0]: Instruction
Instruction to be sent to the external SPI device
This field can be written only when BUSY = 0.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
\begin{tabular}{cccccccccccccccc}
\textbf{ALTERNATE}[31:16] \\

15 & 14 & 13 & 12 & 11 & 10 &  9 &  8 &  7 &  6 &  5 &  4 &  3 &  2 &  1 &  0 \\
\end{tabular}

Bits 31:0 \textbf{ALTERNATE}[31:0]: Alternate bytes
Optional data to be send to the external SPI device right after the address.
This field can be written only when BUSY = 0.

24.5.18 OCTOSPI low-power timeout register (OCTOSPI_LPTR)

Address offset: 0x00130
Reset value: 0x0000 0000

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
\begin{tabular}{cccccccccccccccc}
\textbf{TIMEOUT}[15:0] \\

15 & 14 & 13 & 12 & 11 & 10 &  9 &  8 &  7 &  6 &  5 &  4 &  3 &  2 &  1 &  0 \\
\end{tabular}

Bits 15:0 \textbf{TIMEOUT}[15:0]: Timeout period
After each access in Memory-mapped mode, the OCTOSPI prefetches the subsequent bytes and hold them in the FIFO.
This field indicates how many CLK cycles the OCTOSPI waits after the clock becomes inactive and until it raises the nCS, putting the external device in a lower-consumption state.
This field can be written only when BUSY = 0.
## OCTOSPI wrap communication configuration register (OCTOSPI_WPCCR)

Address offset: 0x0140  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-30</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 29</td>
<td><strong>DQSE</strong>: DQS enable</td>
</tr>
<tr>
<td></td>
<td>This bit enables the data strobe management.</td>
</tr>
<tr>
<td></td>
<td>0: DQS disabled</td>
</tr>
<tr>
<td></td>
<td>1: DQS enabled</td>
</tr>
<tr>
<td>Bit 28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 27</td>
<td><strong>DDTR</strong>: Data double transfer rate</td>
</tr>
<tr>
<td></td>
<td>This bit sets the DTR mode for the data phase.</td>
</tr>
<tr>
<td></td>
<td>0: DTR mode disabled for data phase</td>
</tr>
<tr>
<td></td>
<td>1: DTR mode enabled for data phase</td>
</tr>
<tr>
<td></td>
<td>This field can be written only when BUSY = 0.</td>
</tr>
<tr>
<td>Bits 26:24</td>
<td><strong>DMODE[2:0]</strong>: Data mode</td>
</tr>
<tr>
<td></td>
<td>This field defines the Data phase’s mode of operation.</td>
</tr>
<tr>
<td></td>
<td>000: No data</td>
</tr>
<tr>
<td></td>
<td>001: Data on a single line</td>
</tr>
<tr>
<td></td>
<td>010: Data on two lines</td>
</tr>
<tr>
<td></td>
<td>011: Data on four lines</td>
</tr>
<tr>
<td></td>
<td>100: Data on eight lines</td>
</tr>
<tr>
<td></td>
<td>101-111: Reserved</td>
</tr>
<tr>
<td></td>
<td>This field can be written only when BUSY = 0.</td>
</tr>
<tr>
<td>Bits 23:22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 21:20</td>
<td><strong>ABSIZE[1:0]</strong>: Alternate bytes size</td>
</tr>
<tr>
<td></td>
<td>This bit defines alternate bytes size.</td>
</tr>
<tr>
<td></td>
<td>00: 8-bit alternate bytes</td>
</tr>
<tr>
<td></td>
<td>01: 16-bit alternate bytes</td>
</tr>
<tr>
<td></td>
<td>10: 24-bit alternate bytes</td>
</tr>
<tr>
<td></td>
<td>11: 32-bit alternate bytes</td>
</tr>
<tr>
<td></td>
<td>This field can be written only when BUSY = 0.</td>
</tr>
<tr>
<td>Bit 19</td>
<td><strong>ABDTR</strong>: Alternate bytes double transfer rate</td>
</tr>
<tr>
<td></td>
<td>This bit sets the DTR mode for the alternate bytes phase.</td>
</tr>
<tr>
<td></td>
<td>0: DTR mode disabled for alternate bytes phase</td>
</tr>
<tr>
<td></td>
<td>1: DTR mode enabled for alternate bytes phase</td>
</tr>
<tr>
<td></td>
<td>This field can be written only when BUSY = 0.</td>
</tr>
</tbody>
</table>
Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This field defines the Alternate byte phase’s mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
101-111: Reserved
This field can be written only when BUSY = 0.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This field defines address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address
This field can be written only when BUSY = 0.

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for address phase
1: DTR mode enabled for address phase
This field can be written only when BUSY = 0.

Bits 10:8 **ADMODE[2:0]**: Address mode
This field defines the Address phase’s mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
101-111: Reserved
This field can be written only when BUSY = 0.

Bits 7:6 Reserved, must be kept at reset value.
Bits 5:4 **ISIZE[1:0]: Instruction size**  
This field defines instruction size.  
00: 8-bit instruction  
01: 16-bit instruction  
10: 24-bit instruction  
11: 32-bit instruction  
This field can be written only when BUSY = 0.

Bit 3 **IDTR: Instruction double transfer rate**  
This bit sets the DTR mode for the instruction phase.  
0: DTR mode disabled for instruction phase  
1: DTR mode enabled for instruction phase  
This field can be written only when BUSY = 0.

Bits 2:0 **IMODE[2:0]: Instruction mode**  
This field defines the Instruction phase’s mode of operation.  
000: No instruction  
001: Instruction on a single line  
010: Instruction on two lines  
011: Instruction on four lines  
100: Instruction on eight lines  
101-111: Reserved  
This field can be written only when BUSY = 0.

## 24.5.20 OCTOSPI wrap timing configuration register (OCTOSPI_WPTCR)

Address offset: 0x0148  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>SSHIFT: Sample shift</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>By default, the OCTOSPI samples data 1/2 of a CLK cycle after the data is driven by the external device. This bit allows the data to be sampled later in order to consider the external signal delays. 0: No shift 1: 1/2 cycle shift</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>The firmware must assure that SSHIFT=0 when the data phase is configured in DTR mode (when DDTR = 1). This field can be modified only when BUSY = 0.</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
</tbody>
</table>
24.5.21  OCTOSPI wrap instruction register (OCTOSPI_WPIR)

Address offset: 0x0150
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTRUCTION[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0  **INSTRUCTION[31: 0]:** Instruction
- Instruction to be sent to the external SPI device
- This field can be written only when BUSY = 0.

24.5.22  OCTOSPI wrap alternate bytes register (OCTOSPI_WPABR)

Address offset: 0x0160
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTERNATE[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0  **ALTERNATE[31: 0]:** Alternate bytes
- Optional data to be sent to the external SPI device right after the address
- This field can be written only when BUSY = 0.
### OCTOSPI write communication configuration register (OCTOSPI_WCCR)

Address offset: 0x0180  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Field</th>
<th>Accessibility</th>
<th>Bits 31-30</th>
<th>Bits 29</th>
<th>Bits 28</th>
<th>Bits 27</th>
<th>Bits 26-24</th>
<th>Bits 23-22</th>
<th>Bits 21-20</th>
<th>Bits 19</th>
<th>Bits 18</th>
<th>Bits 17-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DQSE</td>
<td>rw</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| DDTR  | rw            | Bit 29: **DQSE**: DQS enable  
This bit enables the data strobe management.  
0: DQS disabled  
1: DQS enabled |         |         |         |             |             |             |             |         |         |             |
| DMODE[2:0] | rw | Bit 28: Reserved, must be kept at reset value. |         |         |         |             |             |             |         |         |             |
| ABDTR | rw            | Bit 27: **DDTR**: data double transfer rate  
This bit sets the DTR mode for the data phase.  
0: DTR mode disabled for data phase  
1: DTR mode enabled for data phase  
This field can be written only when BUSY = 0. |         |         |         |             |             |             |             |         |         |             |
| ABSIZE[1:0] | rw | Bits 26-24: **DMODE[2:0]**: Data mode  
This field defines the Data phase's mode of operation.  
000: No data  
001: Data on a single line  
010: Data on two lines  
011: Data on four lines  
100: Data on eight lines  
101-111: Reserved  
This field can be written only when BUSY = 0. |         |         |         |             |             |             |             |         |         |             |
| ABDTR | rw            | Bits 23-22: Reserved, must be kept at reset value. |         |         |         |             |             |             |         |         |             |
| ABSIZE[1:0] | rw | Bits 21-20: **ABSIZE[1:0]**: Alternate bytes size  
This field defines alternate bytes size:  
00: 8-bit alternate bytes  
01: 16-bit alternate bytes  
10: 24-bit alternate bytes  
11: 32-bit alternate bytes  
This field can be written only when BUSY = 0. |         |         |         |             |             |             |             |         |         |             |
| ABDTR | rw            | Bit 19: **ABDTR**: Alternate bytes double transfer rate  
This bit sets the DTR mode for the alternate bytes phase.  
0: DTR mode disabled for alternate bytes phase  
1: DTR mode enabled for alternate bytes phase  
This field can be written only when BUSY = 0. |         |         |         |             |             |             |             |         |         |             |
Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This field defines the Alternate byte phase's mode of operation.
- 000: No alternate bytes
- 001: Alternate bytes on a single line
- 010: Alternate bytes on two lines
- 011: Alternate bytes on four lines
- 100: Alternate bytes on eight lines
- 101-111: Reserved
This field can be written only when BUSY = 0.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This field defines address size.
- 00: 8-bit address
- 01: 16-bit address
- 10: 24-bit address
- 11: 32-bit address
This field can be written only when BUSY = 0.

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
- 0: DTR mode disabled for address phase
- 1: DTR mode enabled for address phase
This field can be written only when BUSY = 0.

Bits 10:8 **ADMODE[2:0]**: Address mode
This field defines the Address phase’s mode of operation.
- 000: No address
- 001: Address on a single line
- 010: Address on two lines
- 011: Address on four lines
- 100: Address on eight lines
- 101-111: Reserved
This field can be written only when BUSY = 0.

Bits 7:6 Reserved, must be kept at reset value.
Bits 5:4 **ISIZE[1:0]:** Instruction size
This bit defines instruction size:
- 00: 8-bit instruction
- 01: 16-bit instruction
- 10: 24-bit instruction
- 11: 32-bit instruction
This field can be written only when BUSY = 0.

Bit 3 **IDTR:** Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
- 0: DTR mode disabled for instruction phase
- 1: DTR mode enabled for instruction phase
This field can be written only when BUSY = 0.

Bits 2:0 **IMODE[2:0]:** Instruction mode
This field defines the Instruction phase’s mode of operation.
- 000: No instruction
- 001: Instruction on a single line
- 010: Instruction on two lines
- 011: Instruction on four lines
- 100: Instruction on eight lines
- 101-111: Reserved
This field can be written only when BUSY = 0.

### OCTOSPI write timing configuration register (OCTOSPI_WTCR)

Address offset: 0x0188
Reset value: 0x0000 0000

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

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

**DCYC[4:0]:** Number of dummy cycles
This field defines the duration of the dummy phase.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31). It is recommended to have at least 5 dummy cycles when using memories with DQS activated.
This field can be written only when BUSY = 0.

Bits 31:5 **Reserved, must be kept at reset value.**

Bits 4:0 **DCYC[4:0]:** Number of dummy cycles
This field defines the duration of the dummy phase.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31). It is recommended to have at least 5 dummy cycles when using memories with DQS activated.
This field can be written only when BUSY = 0.
### 24.5.25 OCTOSPI write instruction register (OCTOSPI_WIR)

Address offset: 0x0190  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Bits 15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction</td>
<td>Instruction</td>
</tr>
</tbody>
</table>

Optional data to be sent to the external SPI device  
This field can be written only when BUSY = 0.

### 24.5.26 OCTOSPI write alternate bytes register (OCTOSPI_WABR)

Address offset: 0x01A0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Bits 15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternate</td>
<td>Alternate</td>
</tr>
</tbody>
</table>

Optional data to be sent to the external SPI device right after the address  
This field can be written only when BUSY = 0.

### 24.5.27 OCTOSPI HyperBus latency configuration register (OCTOSPI_HLCR)

Address offset: 0x0200  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Bits 15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRWR[7:0]</td>
<td>TACC[7:0]</td>
</tr>
</tbody>
</table>

912/2974
OCTOSPI register map

The following table summarizes the OCTOSPI registers. Refer to the register boundary addresses table for the OCTOSPI register base address.

| Offset | Register          | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|-------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x0000 | OCTOSPI_CR        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value       | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x0004 | Reserved          |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0008 | OCTOSPI_DCR1      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value       | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x000C | OCTOSPI_DCR2      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value       | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x0010 | OCTOSPI_DCR3      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value       | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x0014 | OCTOSPI_DCR4      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value       | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x0018 | Reserved          |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x001C | Reserved          |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

24.5.28 OCTOSPI register map

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 TRWR[7:0]: Read write recovery time
Device read write recovery time expressed in number of communication clock cycles

Bits 15:8 TACC[7:0]: Access time
Device access time expressed in number of communication clock cycles

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

Bit 1 WZL: Write zero latency
This bit enables zero latency on write operations.
0: Latency on write accesses
1: No latency on write accesses

Bit 0 LM: Latency mode
This bit selects the Latency mode.
0: Variable initial latency
1: Fixed latency
### Table 179. OCTOSPI register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Field Width</th>
<th>Field Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0020</td>
<td>OCTOSPI_SR</td>
<td>32</td>
<td>FLEVEL[5:0]</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BUS</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TOF</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SMF</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TFF</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TCF</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TEF</td>
<td>00000000</td>
</tr>
<tr>
<td>0x0024</td>
<td>OCTOSPI_FCR</td>
<td>32</td>
<td>OCTOSPI_FCR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0028</td>
<td>OCTOSPI_DLR</td>
<td>32</td>
<td>DL[31:0]</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0040</td>
<td>OCTOSPI_DLR</td>
<td>32</td>
<td>ADDRESS[31:0]</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0044</td>
<td>OCTOSPI_AR</td>
<td>32</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0048</td>
<td>OCTOSPI_DR</td>
<td>32</td>
<td>DATA[31:0]</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0050</td>
<td>OCTOSPI_AR</td>
<td>32</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0080</td>
<td>OCTOSPI_PSMKR</td>
<td>32</td>
<td>MASK[31:0]</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0084</td>
<td>OCTOSPI_PSMAR</td>
<td>32</td>
<td>MATCH[31:0]</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0090</td>
<td>OCTOSPI_PIR</td>
<td>32</td>
<td>INTERVAL[15:0]</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0094</td>
<td>OCTOSPI_PIR</td>
<td>32</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0100</td>
<td>OCTOSPI_CCR</td>
<td>32</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0104</td>
<td>OCTOSPI_CCR</td>
<td>32</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0108</td>
<td>OCTOSPI_TCR</td>
<td>32</td>
<td>DCYC[4:0]</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Reset value: 00 00
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Bit Field Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x010C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0110</td>
<td>OCTOSPI IR</td>
<td>INSTRUCTION[31:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0114 - 0x011C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0120</td>
<td>OCTOSPI ABR</td>
<td>ALTERNATE[31:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0124 - 0x012C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0130</td>
<td>OCTOSPI LPTR</td>
<td>TIMEOUT[15:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0134 - 0x013C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0140</td>
<td>OCTOSPI_WPCCR</td>
<td>DORSE</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0144</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0148</td>
<td>OCTOSPI_WPTCR</td>
<td>SHIFT</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x014C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0150</td>
<td>OCTOSPI_WPIR</td>
<td>INSTRUCTION[31:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0154 - 0x015C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0160</td>
<td>OCTOSPI_WPABR</td>
<td>ALTERNATE[31:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0164 - 0x017C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0180</td>
<td>OCTOSPI_WCCR</td>
<td>DORSE</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0184</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0188</td>
<td>OCTOSPI_WTCR</td>
<td>DCYC[4:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x018C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 179. OCTOSPI register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0190</td>
<td>OCTOSPI_WIR</td>
<td>INSTRUCTION[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0194 - 0x019C</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x01A0</td>
<td>OCTOSPI_WABR</td>
<td>ALTERNATE[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x01A4 - 0x01FC</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0200</td>
<td>OCTOSPI_HLCR</td>
<td>TRWR[7:0] TACC[7:0] WZL LM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
25 OCTOSPI I/O manager (OCTOSPIM)

25.1 Introduction

The OCTOSPI I/O manager is a low-level interface that enables:
- An efficient OCTOSPI pin assignment with a full I/O Matrix (before alternate function map)
- Multiplex of single/dual/quad/octal SPI interfaces over the same bus

25.2 OCTOSPIM main features

- Supports up to two single/dual/quad/octal SPI Interfaces
- Supports up to 2 ports for pin assignment
- Fully programmable I/O matrix for pin assignment by function (data/control/clock)

25.3 OCTOSPIM functional description

25.3.1 OCTOSPIM block diagram

The block diagram of the OCTOSPI I/O manager is shown in Figure 152.

---

1. The number of ports (n) is 2.
25.3.2 OCTOSPIM matrix

The OCTOSPI I/O manager matrix allows the user to set a fully programmable pre-mapping of functions:

- Any OCTOSPIM_Pn_CLK / OCTOSPIM_Pn_NCLK pair can be mapped independently to OCTOSPI1_CLK/OCTOSPI1_NCLK or OCTOSPI2_CLK/OCTOSPI2_NCLK
- Any OCTOSPIM_Pn_DQS can be mapped independently to OCTOSPI1_DQS or OCTOSPI2_DQS
- Any OCTOSPIM_Pn_NCS can be mapped independently to OCTOSPI1_NCS or OCTOSPI2_NCS
- Any OCTOSPIM_Pn.IO[3:0] and OCTOSPIM_Pn.IO[7:4] can be mapped independently to OCTOSPI1.IO[3:0], OCTOSPI1.IO[7:4], OCTOSPI2.IO[1:0] or OCTOSPI2.IO[7:4]

For each OCTOSPI I/O manager port, individual signal enables and mapping are configured through the corresponding OCTOSPI I/O manager Port n configuration register (OCTOSPIM_PnCR).

When several I/O pins have the same configuration, and are enabled at the same time, the result may not be predictable.

In the default Out of reset configuration, all the signals of the OCTOSPI1 and OCTOSPI2 are mapped, respectively, on Port 1 and on Port 2.

The I/O matrix shall be configured when the OCTOSPIs are disabled, to avoid unexpected transactions on the bus.
25.3.3 OCTOSPIM multiplexer

When the Multiplexed mode is set, the two OCTOSPIs are multiplexed over the same bus. Both OCTOSPIs get the ownership of the bus in turn through a request/acknowledge protocol with REQ/ACK signals.

The multiplexing is enabled by setting the MUXEN bit of the OCTOSPI I/O manager configuration register (OCTOSPIM_CR).

The fairness counter (MAXTRAN) of each OCTOSPI can be used to accurately manage the maximum duration for which a given OCTOSPI is taking the bus: this feature ensures a maximum bus access latency for the other OCTOSPIs. When the bus is released by one OCTOSPI, an arbitration phase occurs, which is round-robin: if the other OCTOSPI requests the bus, it gets it.

OCTOSPIn_nCS are not part of the multiplexing. Only the OCTOSPIn_IO[7:0], the OCTOSPIn_DQS and OCTOSPIn_CLK / OCTOSPIn_NCLK are muxed.

When the Multiplexed mode is used, only clock mode 0 is supported on the OCTOSPIs.

Due to arbitration and bus sharing, the auto polling interval time of the OCTOSPI, when used, may be increased.

Minimum switching duration

The minimum number of cycles needed to switch from an OCTOSPI to another can be configured.

This internal timer guarantees a latency between the falling edge of the REQ signal of the active OCTOSPI, and the rising edge of the ACK signal of the requesting OCTOSPI.

The duration is defined by the REQ2ACK_TIME fields of the OCTOSPI I/O manager configuration register (OCTOSPIM_CR).

Pin mapping in Multiplexed mode

In Multiplexed mode, the mapping of the bus is done as described below:

- OCTOSP1_nCS and OCTOSP2_NCS work in the same way, then in Non-multiplexed mode they have to be assigned to their respective OCTOSPIM_Pn_NCS.
- All the other signals are seen by the I/O matrix as if they were seen from OCTOSP1.

The Multiplexed mode shall be configured when the OCTOSPIs are disabled to avoid unexpected transactions on the bus.
25.4 OCTOSPI I/O manager registers

25.4.1 OCTOSPI I/O manager control register (OCTOSPIM_CR)

Address offset: 0x0000  
Reset value: 0x0000 0000

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

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:16  **REQ2ACK_TIME [7:0]**: REQ to ACK time

In Multiplexed mode (MUXEN = 1), this field defines the time between two transactions.
The value is the number of OCTOSPI clock cycles - 1

Bits 15:1  Reserved, must be kept at reset value.

Bit 0  **MUXEN**: Multiplexed mode enable

This bit enables the multiplexing of the two OCTOSPIs.
0: No multiplexing
1: OCTOSPI1 and OCTOSPI2 are multiplexed over the same bus

25.4.2 OCTOSPI I/O manager Port n configuration register (OCTOSPIM_PnCR) (n=1 to 2)

Address offset: 0x0000 + 0x04*n (n=1 to 2)

Reset value: Block 1: 0x03010111
Reset value: Block 2: 0x07050333

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

**NCSSRC**: NCE select

0: NCE is selected
1: NCE is not selected

**DQSSRC**: DQ select

0: DQ is selected
1: DQ is not selected

**CLKSRC**: Clock select

0: External clock
1: Internal clock

**CLKEN**: Clock enable

0: Clock enabled
1: Clock disabled
Bits 31:27  Reserved, must be kept at reset value.

Bits 26:25  **IOHSRC[1:0]: IO[7:4] Source for Port n**  
This bits select the source of Port n IO[7:4].  
00: OCTOSPI1_IO[3:0] in non muxed mode / MUXED_IO[3:0] in muxed mode  
10: OCTOSPI2_IO[3:0] in non muxed mode / unused in muxed mode  
11: OCTOSPI2_IO[7:4] in non muxed mode / unused in muxed mode

Bit 24  **IOHEN: IO[7:4] Enable for Port n**  
This bit enables the Port n IO[7:4].  
0: IO[7:4] for Port n is disabled  
1: IO[7:4] for Port n is enabled

Bits 23:19  Reserved, must be kept at reset value.

Bits 18:17  **IOLSRC[1:0]: IO[3:0] Source for Port n**  
This bits select the source of Port n IO[3:0].  
00: OCTOSPI1_IO[3:0] in non muxed mode / MUXED_IO[3:0] in muxed mode  
10: OCTOSPI2_IO[3:0] in non muxed mode / unused in muxed mode  
11: OCTOSPI2_IO[7:4] in non muxed mode / unused in muxed mode

Bit 16  **IOLEN: IO[3:0] Enable for Port n**  
This bit enables the Port n IO[3:0].  
0: IO[3:0] for Port n is disabled  
1: IO[3:0] for Port n is enabled

Bits 15:10  Reserved, must be kept at reset value.

Bit 9  **NCSSRC: nCS Source for Port n**  
This bit selects the source of Port n nCS.  
0: OCTOSPI1_nCS  
1: OCTOSPI2_nCS

Bit 8  **NCSEN: nCS Enable for Port n**  
This bit enables the Port n nCS.  
0: nCS for Port n is disabled  
1: nCS for Port n is enabled

Bits 7:6  Reserved, must be kept at reset value.

Bit 5  **DQSSRC: DQS Source for Port n**  
This bit selects the source of Port n DQS.  
0: OCTOSPI1_DQS in non muxed mode / MUXED_DQS in muxed mode  
1: OCTOSPI2_DQS in non muxed mode / unused port in muxed mode

Bit 4  **DQSEN: DQS Enable for Port n**  
This bit enables the Port n DQS.  
0: DQS for Port n is disabled  
1: DQS for Port n is enabled
Bits 3:2  Reserved, must be kept at reset value.

Bit 1  **CLKSRC**: CLK/CLKn Source for Port n
This bit selects the source of Port n CLK/CLKn.
0: OCTOSPI1_CLK/CLKn in non muxed mode / MUXED_CLK/CLKn in muxed mode
1: OCTOSPI2_CLK/CLKn in non muxed mode / unused port in muxed mode

Bit 0  **CLKEN**: CLK/CLKn Enable for Port n
This bit enables the Port n CLK/CLKn.
0: CLK/CLKn for Port n is disabled
1: CLK/CLKn for Port n is enabled
### 25.4.3 OCTOSPIM register map

The following table summarizes the OCTOSPIM I/O manager registers. Refer to the register boundary addresses table for the OCTOSPIM I/O manager register base address.

#### Table 180. OCTOSPIM register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>OCTOSPIM_CR</td>
<td>0x0004</td>
<td>OCTOSPIM_P1CR</td>
<td>0x0008</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td>IOHSRC [1:0]</td>
<td>IOHSRC [1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td>IDHEN</td>
<td>IDHEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td>IQSRC [1:0]</td>
<td>IQSRC [1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td>IQLEN</td>
<td>IQLEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td>NCSSRC</td>
<td>NCSSRC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td>NCSEN</td>
<td>NCSEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td>DQSSRC</td>
<td>DQSSRC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td>DQSEN</td>
<td>DQSEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td>CLKSRC</td>
<td>CLKSRC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td>CLKEN</td>
<td>CLKEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td>MUXEN</td>
<td>MUXEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td>REQ2ACK_TIME[7:0]</td>
<td>RES.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td></td>
<td>RES.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td></td>
<td>RES.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td></td>
<td>RES.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td></td>
<td>RES.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td></td>
<td>RES.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td></td>
<td>RES.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td></td>
<td>RES.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td></td>
<td>RES.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RES.</td>
<td></td>
<td>RES.</td>
</tr>
</tbody>
</table>

Refer to *Section 2.3 on page 128* for the register boundary addresses.
26 Delay block (DLYB)

26.1 Introduction

The delay block (DLYB) is used to generate an output clock which is dephased from the input clock. The phase of the output clock must be programmed by the user application. The output clock is then used to clock the data received by another peripheral such as an SDMMC or Octo-SPI interface.

The delay is voltage- and temperature-dependent, which may require the application to reconfigure and recenter the output clock phase with the receive data.

26.2 DLYB main features

The delay block has the following features:
- Input clock frequency ranging from 25 to 208 MHz
- Up to 12 oversampling phases.

26.3 DLYB functional description

26.3.1 DLYB diagram

The delay block includes the following sub-blocks:
- Register interface block providing AHB access to the Delay Block registers.
- Delay line supporting the unit delays.
- Delay line length sampling
- Output clock selection multiplexer

![Figure 153. DLYB block diagram](image)
26.3.2 DLYB pins and internal signals

*Table 181* lists the DLYB internal signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dlyb_hclk</td>
<td>Digital input</td>
<td>Delay block register interface clock</td>
</tr>
<tr>
<td>dlyb_in_ck</td>
<td>Digital input</td>
<td>Delay block input clock</td>
</tr>
<tr>
<td>dlyb_out_ck</td>
<td>Digital output</td>
<td>Delay block output clock</td>
</tr>
</tbody>
</table>

26.3.3 General description

The delay block is enabled by setting the DEN bit in the DLYB_CR register (see *Section 26.4.1: DLYB control register (DLYB_CR)*). The length sampler is enabled through the SEN bit in DLYB_CR register.

When the delay block is enabled, the delay added by a unit delay is defined by the UNIT bits in DLYB_CFGR register (see *Section 26.4.2: DLYB configuration register (DLYB_CFGR)*). Note that the UNIT bits can be programmed only when the output clock is disabled (SEN = ‘1’).

When the delay block is enabled, the output clock phase is selected through the SEL bit in DLYB_CFGR register. Note that SEL can be programmed only when the output clock is disabled (SEN = ‘1’).

Before dephasing the output clock, the delay line length shall be configured to one input clock period. The delay line length can be configured by enabling the length sampler through the SEN bit, which gives access to the delay line length (LNG bits) and Length valid flag (LNGF) in DLYB_CFGR.

Once the delay line length has been configured, a dephased output clock can be selected by the output clock multiplexer. This is done through SEL bits. The output clock is only available on the selected phase when SEN is set to ‘0’.

*Table 182.* gives a summary of the delay block control.

<table>
<thead>
<tr>
<th>DEN</th>
<th>SEN</th>
<th>UNIT</th>
<th>SEL</th>
<th>LNG</th>
<th>LNGF</th>
<th>Output clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Enabled (= Input clock)</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
<td>Unit delay</td>
<td>Output clock phase</td>
<td>Length</td>
<td>Length flag</td>
<td>Disabled</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Unit delay(1)</td>
<td>Output clock phase(2)</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Enabled (= selected phase)</td>
</tr>
</tbody>
</table>

1. The unit delay can only be changed when SEN = ‘1’.
2. The output clock phase can only be changed when SEN = ‘1’.
26.3.4 Delay line length configuration procedure

LNG bits are used to determine the delay line length with respect to the input clock period. The length shall be configured so that one full input clock period is covered by the delay line length.

To configure the delay line length to one period of the Input clock, follow the sequence below:

1. Enable the delay block by setting DEN bit to ‘1’.  
2. Enable the length sampling by setting SEN bit to ‘1’.  
3. Enable all delay cells by setting SEL bits to 12.  
4. For UNIT = 0 to 127 (this step must be repeated until the delay line length is configured):
   a) Update the UNIT value and wait till the length flag LNGF is set to ‘1’.  
   b) Read LNG bits.  
      If (LNG[10:0] > 0) and (LNG[11] or LNG[10] = 0), the delay line length is configured to one input clock period.
5. Determine how many unit delays (N) span one input clock period.  
   - For N = 10 to 0:  
      If LNG[N] = ‘1’, the number of unit delays spanning the input clock period = N.
6. Disable the length sampling by clearing SEN to ‘0’.

26.3.5 Output clock phase configuration procedure

When the delay line length is configured to one input clock period, the output clock phase can be selected between the unit delays spanning one Input clock period.

Follow the steps below to select the output clock phase:
1. Disable the output clock and enable the access to the phase selection SEL bits by setting SEN bit to ‘1’.  
2. Program SEL bits with the desired output clock phase value.  
3. Enable the output clock on the selected phase by clearing SEN to ‘0’.
26.4  DLYB registers

All registers can be accessed in word, half-word and byte access.

26.4.1  DLYB control register (DLYB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

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

Bit 0  DEN: Delay block enable bit

0: Delay block disabled.
1: Delay block enabled.

26.4.2  DLYB configuration register (DLYB_CFGFR)

Address offset: 0x004

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Length valid flag</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>Delay line length value</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31  LNGF: Length valid flag

This flag indicates when the delay line length value contained in LNG bits is valid after UNIT bits changed.
0: Length value in LNG is not valid.
1: Length value in LNG is valid.

Bits 30:28  Reserved, must be kept at reset value.

Bits 27:16  LNG[11:0]: Delay line length value

These bits reflect the 12 unit delay values sampled at the rising edge of the input clock.
The value is only valid when LNGF = ‘1’.
Bit 15  Reserved, must be kept at reset value.

Bits 14:8  **UNIT[6:0]:** Delay Defines the delay of a Unit delay cell.
            These bits can only be written when SEN = '1'.
            Unit delay = Initial delay + UNIT x delay step

Bits 7:4  Reserved, must be kept at reset value.

Bits 3:0  **SEL[3:0]:** Select the phase for the Output clock.
            These bits can only be written when SEN = '1'.
            Output clock phase = Input clock + SEL x Unit delay

### 26.4.3 DLYB register map

#### Table 183. DLYB register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | DLYB_CR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x004  | DLYB_CFGR     | LNG|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to *Section 2.3 on page 128* for the register boundary addresses.
27 Analog-to-digital converters (ADC)

27.1 Introduction

This section describes the ADC implementation:

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).

Each ADC consists of a 16-bit successive approximation analog-to-digital converter.

Each ADC has up to 20 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 32-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows to improve analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.
27.2 ADC main features

- **High-performance features**
  - Up to 2x ADCs which can operate in dual mode
  - 16, 14, 12, 10 or 8-bit configurable resolution
  - ADC conversion time is independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Can manage Single-ended or differential inputs (programmable per channels)
  - AHB slave bus interface to allow fast data handling
  - Self-calibration (both offset and linearity)
  - Channel-wise programmable sampling time
  - Up to four injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by GP-DMA for regular channel conversions with FIFO
  - Data can be routed to DFSDM for post processing
  - 4 dedicated data registers for the injected channels

- **Oversampler**
  - 32-bit data register
  - Oversampling ratio adjustable from 2 to 1024x
  - Programmable data right and left shift

- **Low-power features**
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (auto-delayed mode)

- **Each ADC features an external analog input channel**
  - Up to 6 fast channels from dedicated GPIO pads
  - Up to 14 slow channels from dedicated GPIO pads

- **In addition, there are 6 internal dedicated channels**
  - Internal reference voltage ($V_{REFINT}$)
  - Internal temperature sensor ($V_{SENSE}$)
  - $V_{BAT}$ monitoring channel ($V_{BAT}/4$)
  - Connection to DAC internal channels

- **Start-of-conversion can be initiated:**
  - by software for both regular and injected conversions
  - by hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions

- **Conversion modes**
  - Each ADC can convert a single channel or can scan a sequence of channels
– Single mode converts selected inputs once per trigger
– Continuous mode converts selected inputs continuously
– Discontinuous mode

• Dual ADC mode
• Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
• 3 analog watchdogs per ADC
• ADC input range: \( V_{REF-} \leq V_{IN} \leq V_{REF+} \)

*Figure 154* shows the block diagram of one ADC.

### 27.3 ADC implementation

<table>
<thead>
<tr>
<th>ADC modes/features</th>
<th>ADC1</th>
<th>ADC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual mode</td>
<td>X (coupled together)</td>
<td></td>
</tr>
<tr>
<td>DFSDM interface</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Extended sample time option (SMPPLUS control)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LDO voltage regulator status</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
27.4 ADC functional description

27.4.1 ADC block diagram

Figure 154 shows the ADC block diagram and Table 185 gives the ADC pin description.
27.4.2 ADC pins and internal signals

Table 185. ADC input/output pins

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREF+</td>
<td>Input, analog reference positive</td>
<td>The higher/positive reference voltage for the ADC.</td>
</tr>
<tr>
<td>VDDA</td>
<td>Input, analog supply</td>
<td>Analog power supply equal $V_{DDA}$: $1.62 , V \leq V_{DDA} \leq 3.6 , V$</td>
</tr>
<tr>
<td>VREF-</td>
<td>Input, analog reference negative</td>
<td>The lower/negative reference voltage for the ADC.</td>
</tr>
<tr>
<td>VSSA</td>
<td>Input, analog supply ground</td>
<td>Ground for analog power supply equal to $V_{SS}$</td>
</tr>
<tr>
<td>ADCx_INPy</td>
<td>External analog inputs</td>
<td>Up to 20 analog input channels ($x = \text{ADC number} = 1$ to $2$):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ADCx_INP[0:5] fast channels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ADCx_INP[6:19] slow channels</td>
</tr>
<tr>
<td>ADCx_INNy</td>
<td>External analog inputs</td>
<td>Up to 20 analog input channels ($x = \text{ADC number} = 1$ to $2$):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ADCx_INN[0:5] fast channels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ADCx_INN[6:19] slow channels</td>
</tr>
</tbody>
</table>

Table 186. ADC internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{INP}[y]$</td>
<td>Analog inputs</td>
<td>Positive input analog channels for each ADC, connected either to ADCx_INP/ external channels or to internal channels.</td>
</tr>
<tr>
<td>$V_{INN}[y]$</td>
<td>Analog inputs</td>
<td>Negative input analog channels for each ADC, connected either to $V_{REF-}$ or to ADCx_INN/ external channels</td>
</tr>
<tr>
<td>adc_ext_trgy</td>
<td>Inputs</td>
<td>Up to 21 external trigger inputs for the regular conversions (can be connected to on-chip timers). These inputs are shared between the ADC master and the ADC slave.</td>
</tr>
<tr>
<td>adc_jext_trgy</td>
<td>Inputs</td>
<td>Up to 21 external trigger inputs for the injected conversions (can be connected to on-chip timers). These inputs are shared between the ADC master and the ADC slave.</td>
</tr>
<tr>
<td>adc_awd1, adc_awd2, adc_awd3</td>
<td>Outputs</td>
<td>Internal analog watchdog output signal connected to on-chip timers. ($x = \text{Analog watchdog number} 1,2,3$)</td>
</tr>
<tr>
<td>adc_it</td>
<td>Output</td>
<td>ADC interrupt</td>
</tr>
<tr>
<td>adc_hclk</td>
<td>Input</td>
<td>AHB clock</td>
</tr>
<tr>
<td>adc_sclk</td>
<td>Input</td>
<td>ADC synchronous clock input from RCC</td>
</tr>
<tr>
<td>adc_ker_ck_input</td>
<td>Input</td>
<td>ADC kernel clock input from RCC</td>
</tr>
</tbody>
</table>
27.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC clock is independent from the AHB bus clock.

The input clock is the same for all ADCs and can be selected between two different clock sources (see Figure 155: ADC Clock scheme):

1. The ADC clock can be a specific clock source, named adc_ker_ck_input which is independent and asynchronous with the AHB clock.
   
   It can be configured in the RCC (refer to RCC Section for more information on how to generate the ADC clock (adc_ker_ck_input) dedicated clock).
   
   To select this scheme, CKMODE[1:0] bits of the ADCx_CCR register must be reset.

2. The ADC clock can be derived from the system clock or system clock divided by two (adc_sclk). In this mode, a programmable divider factor can be selected (/1, 2 or 4 according to bits CKMODE[1:0]).
   
   To select this scheme, CKMODE[1:0] bits of the ADCx_CCR register must be different from "00". adc_sclk is equal to sys_ck when HPRE is set to 0, otherwise it corresponds to sys_ck/2.

   In both case, the clock divider factor of 2 is applied to the clock provided to the ADC analog block (f_{adc_ker_ck}).

Option 1) has the advantage of reaching the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128, 256; using the prescaler configured with bits PRESC[3:0] in the ADCx_CCR register.
Option 2) has the advantage of using the system without additional PLL. In addition, when adc_sclk is twice faster than the adc_hclk clock, the latency between the trigger and the start of conversion is fixed. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).

The clock configured through CKMODE[1:0] bits must be compliant with the analog ADC operating frequency specified in the product datasheet.

Note:  adc_sclk is the system clock or system clock divided by two: when the ABH prescaler is set to 1 (HPRE[3:0] = 0XXX in RCC_CFGR register), adc_sclk is equal to sys_clk, otherwise adc_sclk corresponds to sys_clk/2.

Figure 155. ADC Clock scheme

1. Refer to the RCC section to see how adc_hclk and adc_ker_ck_input can be generated.

Clock ratio constraint between ADC clock and AHB clock

There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- $F_{adc\_hclk} \geq 4 \times F_{adc\_ker\_ck}$ if the resolution of all channels are 16-bit, 14-bit, 12-bit or 10-bit
- $F_{adc\_hclk} \geq 3 \times F_{adc\_ker\_ck}$ if there are some channels with resolutions equal to 8-bit (and none with lower resolutions)

BOOST control

The ADC Boost mode can be controlled through the BOOST bitfield in the ADC_CR register.

This bitfield must be set according to the ADC clock setting. Refer to the ADC_CR register description.
27.4.4 ADC1/2 connectivity

ADC1 and ADC2 are tightly coupled and share some external channels as described in the following figures.

Figure 156. ADC1 connectivity

1. ADCx_INNy signal can only be used when the corresponding ADC input channel is configured as differential mode.
1. ADCx_INNy signal can only be used when the corresponding ADC input channel is configured as differential mode.
27.4.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

27.4.6 ADC deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in deep-power-down mode where its supply is internally switched off to reduce the leakage currents (the reset state of bit DEEPPWD is 1 in the ADC.CR register).

To start ADC operations, it is first needed to exit deep-power-down mode by clearing bit DEEPPWD=0.

Then, it is mandatory to enable the ADC internal voltage regulator by setting the bit ADVREGEN=1 into ADC.CR register. The software must wait for the startup time of the ADC voltage regulator (TADVREG.STUP) before launching a calibration or enabling the ADC. This delay must be implemented by software.

The LDO status can be verified by checking the LDORDY bit in ADC_ISR register (refer to Section 27.3: ADC implementation for the availability of the LDO regulator status).

For the startup time of the ADC voltage regulator, refer to device datasheet for TADVREG.STUP parameter.

After ADC operations are complete, the ADC can be disabled (ADEN=0). It is possible to save power by also disabling the ADC voltage regulator. This is done by writing bit ADVREGEN=0.

Then, to save more power by reducing the leakage currents, it is also possible to re-enter in ADC deep-power-down mode by setting bit DEEPPWD=1 into ADC.CR register. This is particularly interesting before entering Stop mode.

Note: Writing DEEPPWD=1 automatically disables the ADC voltage regulator and bit ADVREGEN is automatically cleared.

Note: When the internal voltage regulator is disabled (ADVREGEN=0), the internal analog calibration is kept.

In ADC deep-power-down mode (DEEPPWD=1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or apply again the calibration factor which was previously saved (refer to Section 27.4.8: Calibration (ADCAL, ADCALDIFF, ADCALLIN, ADC_CALFACT)).
27.4.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by writing into bits DIFSEL[19:0] in the ADC_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN=0).

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage $V_{INP[i]}$ (positive input) and $V_{REF-}$ (negative input).

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage $V_{INP[i]}$ (positive input) and $V_{INN[i]}$ (negative input).

The output data for the differential mode is an unsigned data. When $V_{INP[i]}$ is $V_{REF-}$, $V_{INN[i]}$ is $V_{REF+}$, the output data is 0x0000 (16-bit resolution mode), when $V_{INP[i]}$ is $V_{REF+}$, $V_{INN[i]}$ is $V_{REF-}$, the output data is 0xFFFF.

When ADC is configured as differential mode, both input should be biased at $V_{REF+} / 2$ voltage.

The input signal are supposed to be differential (common mode voltage should be fixed).

For a complete description of how the input channels are connected for each ADC, refer to Section 27.4.4: ADC1/2 connectivity.

Caution: When configuring the channel “i” in differential input mode, its negative input voltage is connected to $V_{INN[i]}$. As a consequence, channel “i+n”, which is connected to $V_{INN[i]}$, should not be converted at same time by different ADCs. Some channels are shared between ADC1/ADC2: this can make the channel on the other ADC unusable.

27.4.8 Calibration (ADCAL, ADCALDIF, ADCALLIN, ADC_CALFACT)

Each ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor which is 11-bits of offset or 160-bits of linearity and which is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

The calibration is preliminary to any ADC operation. It removes the systematic errors which may vary from chip to chip and allows to compensate offset and linearity deviation.

The calibration factor for the offset to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIF=0 before launching a calibration which will be applied for single-ended input conversions.
- Write ADCALDIF=1 before launching a calibration which will be applied for differential input conversions.

\[
\text{Converted value} = \frac{\text{ADC \_ Full \_ Scale}}{2} \times \left[ 1 + \frac{V_{INP} - V_{INN}}{V_{REF+}} \right]
\]
The linearity correction must be done once only, regardless of single / differential configuration.

- Write ADCALLIN=1 before launching a calibration which will run the linearity calibration at the same time as the offset calibration.
- Write ADCALLIN=0 before launching a calibration which will not run the linearity calibration but only the offset calibration.

The calibration is then initiated by software by setting bit ADCAL=1. It can be initiated only when the ADC is disabled (when ADEN=0). ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[10:0] or CALFACT_D[10:0] of ADC_CALFACT register (depending on single-ended or differential input calibration). The 160-bit linearity calibration factor can be accessed using the ADC_CALFACT2 register with ADEN set to 1.

The internal analog calibration is kept if the ADC is disabled (ADEN=0). However, if the ADC is disabled for extended periods, it is recommended that a new offset calibration cycle is run before enabling again the ADC.

The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in STANDBY or VBAT mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADC_CALFACT and ADC_CALFACT2 register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN=1 and ADSTART=0 and JADSTART=0). Then, at the next start of conversion, the calibration factor will automatically be injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when VREF+ voltage changed more than 10%.

Refer to the datasheets for the clock cycle requirement for both linear and offset calibration.

**Software procedure to calibrate the ADC**

1. Ensure DEEPPWD=0, ADVREGEN=1 and verify that the ADC voltage regulator startup time has elapsed by checking the LDORDY bit in ADC_ISR (refer to Section 27.3: ADC implementation for the availability of the LDO regulator status).
2. Ensure that ADEN=0.
3. Select the input mode for this calibration by setting ADCALDIF=0 (Single-ended input) or ADCALDIF=1 (Differential input). Select if Linearity calibration enable or not by ADCALLIN=1 (enabled) or ADCALLIN=0 (disabled).
4. Set ADCAL=1.
5. Wait until ADCAL=0.
6. The offset calibration factor can be read from ADC_CALFACT register.
7. The linearity calibration factor can be read from ADC_CALFACT2 register, following the procedure described in Section : Linearity calibration reading procedure (ADEN must be set to 1 prior to accessing ADC_CALFACT2 register).
Software procedure to re-inject a calibration factor into the ADC

1. Ensure ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new offset calibration factors.
3. Write LINCALFACT bits with the new linearity calibration factors, following the procedure described in Section : Linearity calibration writing procedure.
4. When a conversion is launched, the calibration factor will be injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Figure 158. ADC calibration

Figure 159. Updating the ADC offset calibration factor
Calibrating single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF=0 and one with ADCALDIF=1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF=0) and Linearity calibration enable (with ADCALLIN=1). This updates the registers CALFACT_S[10:0] and LINCALFACT[159:0].
3. Calibrate the ADC in Differential input modes (with ADCALDIF=1) and Linearity calibration disable (with ADCALLIN=0). This updates the register CALFACT_D[10:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration will automatically be injected into the analog ADC.

Figure 160. Mixing single-ended and differential channels

![Diagram](MSv41023V1)
Linearity calibration reading procedure

Once the calibration is done (ADCAL bit cleared by hardware) with ADCALLIN=1, the 160-bit linearity correction factor can be read using the ADC_CALFACT2 30-bit registers (6 read accesses are necessary).

The six LINCALRDYW1..6 control/status bits in ADC_CR are set when the calibration is complete. When ADEN is set to 1, clearing one of these bits launches the transfer of part of the linearity factor into the LINCALFACT[29:0] of the ADC_CALFACT2 register. The bit will be reset by hardware when the ADC_CALFACT2 register can be read (software must poll the bit until it is cleared). The complete procedure is as following:

1. Ensure DEEPPWD=0, ADVREGEN=1 and that the ADC voltage regulator startup time has elapsed by checking the LDORDY bit in ADC_ISR (refer to Section 27.3: ADC implementation for the availability of the LDO regulator status).
2. Set ADEN = 1 and wait until ADRDY=1.
4. Poll LINCALRDYW6 bit until returned value is zero, indicating linearity correction bits[159:150] are available in ADC_CALFACT2[29:0].
5. Read ADC_CALFACT2[29:0].
7. Poll LINCALRDYW5 bit until returned value is zero, indicating linearity correction bits[149:120] are available in ADC_CALFACT2[29:0].
8. Read ADC_CALFACT2[29:0].
10. Poll LINCALRDYW4 bit until returned value is zero, indicating linearity correction bits[119:90] are available in ADC_CALFACT2[29:0].
11. Read ADC_CALFACT2[29:0].
13. Poll LINCALRDYW3 bit until returned value is zero, indicating linearity correction bits[89:60] are available in ADC_CALFACT2[29:0].
14. Read ADC_CALFACT2[29:0].
15. Clear LINCALRDYW2 bit.
16. Poll LINCALRDYW2 bit until returned value is zero, indicating linearity correction bits[59:30] are available in ADC_CALFACT2[29:0].
17. Read ADC_CALFACT2[29:0].
18. Clear LINCALRDYW1 bit.
19. Poll LINCALRDYW1 bit until returned value is zero, indicating linearity correction bits[29:0] are available in ADC_CALFACT2[29:0].
20. Read ADC_CALFACT2[29:0].

Note: The software is allowed to toggle a single LINCALRDYWx bit at once (other bits left unchanged), otherwise causing unexpected behavior.

The software can access the linearity calibration factor by writing LINCALRDYW1..6 bits only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).
Linearity calibration writing procedure

The six LINCALRDYW1..6 control/status bits in ADC_CR are reset when the calibration has not yet been done or a new linearity calibration factor have been rewritten. It is possible to force directly a linearity calibration factor or re-inject it using the following procedure:

1. Ensure DEEPPWD=0, ADVREGEN=1 and that ADC voltage regulator startup time has elapsed by checking the LDORDY bit in ADC_ISR (refer to Section 27.3: ADC implementation for the availability of the LDO regulator status).
2. Set ADEN = 1 and wait until ADRDY=1.
3. Write ADC_CALFACT2[9:0] with previously saved linearity correction factor bits[159:150].
4. Set LINCALRDYW6 bit.
5. Poll LINCALRDYW6 bit until returned value is one, indicating linearity correction bits[159:150] have been effectively written.
6. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[149:120].
7. Set LINCALRDYW5 bit.
8. Poll LINCALRDYW5 bit until returned value is one, indicating linearity correction bits[149:120] have been effectively written.
9. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[119:90].
10. Set LINCALRDYW4 bit.
11. Poll LINCALRDYW4 bit until returned value is one, indicating linearity correction bits[119:90] have been effectively written.
12. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[89:60].
13. Set LINCALRDYW3 bit.
14. Poll LINCALRDYW3 bit until returned value is one, indicating linearity correction bits[89:60] have been effectively written.
15. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[59:30].
17. Poll LINCALRDYW2 bit until returned value is one, indicating linearity correction bits[59:30] have been effectively written.
18. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[29:0].
19. Set LINCALRDYW1 bit.
20. Poll LINCALRDYW1 bit until returned value is one, indicating linearity correction bits[29:0] have been effectively written.

Note: The software is allowed to toggle a single LINCALRDYWx bit at once (other bits left unchanged), otherwise causing unexpected behavior.

The software is allowed to update the linearity calibration factor by writing LINCALRDYW1..6 bits only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).
27.4.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in Section 27.4.6: ADC deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)).

Once DEEPPWD=0 and ADVREGEN=1, the ADC can be enabled and the ADC needs a stabilization time of \( t_{\text{STAB}} \) before it starts converting accurately, as shown in Figure 161. Two control bits enable or disable the ADC:

- **ADEN=1** enables the ADC. The flag ADRDY will be set once the ADC is ready for operation.
- **ADDIS=1** disables the ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART=1 (refer to Section 27.4.19: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART=1 or when an external injected trigger event occurs, if injected triggers are enabled.

**Software procedure to enable the ADC**

1. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’.
2. Set ADEN=1.
3. Wait until ADRDY=1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE=1).
4. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’ (optional).

**Software procedure to disable the ADC**

1. Check that both ADSTART=0 and JADSTART=0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP=1 and JADSTP=1 and then wait until ADSTP=0 and JADSTP=0.
2. Set ADDIS=1.
3. If required by the application, wait until ADEN=0, until the analog ADC is effectively disabled (ADDIS will automatically be reset once ADEN=0).

![Figure 161. Enabling / Disabling the ADC](MSv30264V2)
27.4.10  Constraints when writing the ADC control bits

The software can write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the control bits DIFSEL in the ADC_DIFSEL register, ADCx_CCR register and the control bits ADCAL and ADEN in the ADC_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADC_CFGR, ADC_SMPRy, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if the ADC is enabled (ADEN=1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).

- For control bits related to configuration of injected conversions, the software is allowed to write them only if the ADC is enabled (ADEN=1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).

The software can write ADSTP or JADSTP control bits in the ADC_CR register only if the ADC is enabled and eventually converting and if there is no pending request to disable the ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).

The software can write the register ADC_JSQR at any time, when the ADC is enabled (ADEN=1).

The software is allowed to write the ADC_JSQR register only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register).

Note: There is no hardware protection to prevent these forbidden write accesses and ADC behavior may become in an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN=0 as well as all the bits of ADC_CR register).

27.4.11  Channel selection (SQRx, JSQRx)

There are up to 20 multiplexed channels per ADC:

- 6 fast analog inputs coming from Analog PADs and GPIO pads (ADCx_INP/INN[0..5])
- Up to 14 slow analog inputs coming from GPIO pads (ADCx_INP/INN[6..19]).

The ADCs are connected to 6 internal analog inputs:

- the internal temperature sensor ($V_{SENSE}$)
- the internal reference voltage ($V_{REFINT}$)
- the $V_{BAT}$ monitoring channel ($V_{BAT}/4$)
- DAC internal channels

Refer to Table ADC interconnection in Section 27.4.2: ADC pins and internal signals for the connection of the above internal analog inputs to external ADC pins or internal signals.

It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order:
ADCx_INP/INN3, ADCx_INP/INN8, ADCx_INP/INN2, ADCx_INP/INN0, ADCx_INP/INN2, ADCx_INP/INN2, ADCx_INP/INN15.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.

- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP=1 (refer to Section 27.4.18: Stopping an ongoing conversion (ADSTP, JADSTP)).

The software is allowed to modify on-the-fly the ADC_JSQR register when JADSTART is set to 1 only when the context queue is enabled (JQDIS=0 in ADC_CFR register).

**Temperature sensor, VREFINT and VBAT internal channels**

The internal reference voltage (VREFINT), the temperature sensor (VSENSE), and the VBAT channel are connected to ADC internal channels. Refer to Table ADC interconnection in Section 27.4.2: ADC pins and internal signals for details.

*Note:* To convert one of the internal analog channels, enable the corresponding analog sources by programming VREFEN, TSEN and VBATEN bits in the ADCx_CCR registers.

### 27.4.12 Channel preselection register (ADC_PCSEL)

For each channel selected through SQRx or JSQRx, the corresponding ADC_PCSEL bit must be previously configured.

This ADC_PCSEL bit controls the analog switch integrated in the I/O level. The ADC input MUX selects the ADC input according to the SQRx and JSQRx with very high speed, the analog switch integrated in the IO cannot react as fast as ADC mux does. To avoid the delay on analog switch control on IO, it is necessary to pre select the input channels which will be selected in the SQRx, JSQRx.

The selection is based on the VINP[i] of each ADC input. If ADC1 converts the ADC12_INP2(VINP[2]) as differential mode, ADC12_INP6(VINP[6]) also needs to be selected in ADC_PCSEL.

Some I/Os are connected to several VINP[i] of the ADCx. The control inputs of the analog switch are ORed with the corresponding ADC_PCSEL register bits.
27.4.13 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 1.5 ADC clock cycles
- SMP = 001: 2.5 ADC clock cycles
- SMP = 010: 8.5 ADC clock cycles
- SMP = 011: 16.5 ADC clock cycles
- SMP = 100: 32.5 ADC clock cycles
- SMP = 101: 64.5 ADC clock cycles
- SMP = 110: 387.5 ADC clock cycles
- SMP = 111: 810.5 ADC clock cycles

The total conversion time is calculated as follows:

\[ T_{CONV} = \text{Sampling time} + 7.5 \text{ ADC clock cycles} \]

Example:

With \( F_{\text{adc.ker.clk}} = 24 \text{ MHz} \) and a sampling time of 1.5 ADC clock cycles (14-bit mode):

\[ T_{CONV} = (1.5 + 7.5) \text{ ADC clock cycles} = 9 \text{ ADC clock cycles} = 0.375 \mu \text{s} \]  

(14-bit mode for fast channels)

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

Constraints on the sampling time for fast and slow channels

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the datasheets.

I/O analog switches voltage booster

The I/O analog switches resistance increases when the \( V_{\text{DDA}} \) voltage is too low. This requires to have the sampling time adapted accordingly (refer to the datasheet for electrical characteristics). This resistance can be minimized at low \( V_{\text{DDA}} \) by enabling an internal voltage booster with BOOSTE bit in the PWR_CR1 register.

27.4.14 Single conversion mode (CONT=0)

In Single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADC_CR register (for a regular channel, with software trigger selected)
- Setting the JADSTART bit in the ADC_CR register (for an injected channel, with software trigger selected)
- External hardware trigger event (for a regular or injected channel)

ADSTART bit or JADSTART bit must be set before triggering an external event.
Inside the regular sequence, after each conversion is complete:
- The converted data are stored into the 32-bit ADC_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:
- The converted data are stored into one of the four 32-bit ADC_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:
- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:
- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

Note: To convert a single channel, program a sequence with a length of 1.

27.4.15 Continuous conversion mode (CONT=1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically re-starts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:
- The converted data are stored into the 32-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:
- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to Auto-injection mode section.)
27.4.16 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART=1.

When ADSTART is set, the conversion starts:
- Immediately: if EXTEN = 0x0 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN /= 0x0

Software starts ADC injected conversions by setting JADSTART=1.

When JADSTART is set, the conversion starts:
- Immediately, if JEXTEN = 0x0 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN /= 0x0

Note: In auto-injection mode (JAUTO=1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART=0 and JADSTART=0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:
- In single mode with software trigger (CONT=0, EXTEN=0x0)
  - at any end of conversion sequence (EOS =1)
- In discontinuous mode with software trigger (CONT=0, DISCEN=1, EXTEN=0x0)
  - at end of conversion (EOC=1)
- In all other cases (CONT=x, EXTEN=x)
  - after execution of the ADSTP procedure asserted by the software.

Note: In continuous mode (CONT=1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT=0 and EXTEN /=0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is cleared by hardware:
- in single mode with software injected trigger (JEXTEN=0x0)
  - at any end of injected conversion sequence (JEOS assertion) or at any end of sub-group processing if JDISCEN=1
- in all cases (JEXTEN=x)
  - after execution of the JADSTP procedure asserted by the software.

Note: When the software trigger is selected, ADSTART bit should not be set if the EOC flag is still high.
27.4.17 Timing

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

\[
T_{\text{CONV}} = T_{\text{SMPL}} + T_{\text{SAR}} = [1.5 \text{ min} + 7.5 \text{ 14bit}] \times T_{\text{adc\_ker\_ck}}
\]

\[
T_{\text{CONV}} = T_{\text{SMPL}} + T_{\text{SAR}} = 62.5 \text{ ns } \text{min} + 312.5 \text{ ns } \text{14bit} = 375.0 \text{ ns} \quad \text{(for } F_{\text{adc\_ker\_ck}} = 24 \text{ MHz})
\]

Figure 162. Analog to digital conversion time

<table>
<thead>
<tr>
<th>ADC state</th>
<th>RDY</th>
<th>Sampling Ch(N)</th>
<th>Converting Ch(N)</th>
<th>Sampling Ch(N+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog channel</td>
<td>Ch(N)</td>
<td></td>
<td>Ch(N+1)</td>
<td></td>
</tr>
<tr>
<td>Internal S/H</td>
<td>Sample AIN(N)</td>
<td>Hold AIN(N)</td>
<td>Sample AIN(N+1)</td>
<td></td>
</tr>
<tr>
<td>ADSTART</td>
<td>Set by S/W</td>
<td>Cleared by H/W</td>
<td>Set by S/W</td>
<td></td>
</tr>
<tr>
<td>EOSMP</td>
<td></td>
<td></td>
<td>Cleared by S/W</td>
<td></td>
</tr>
<tr>
<td>EOC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC_DR</td>
<td></td>
<td>Data N-1</td>
<td>Data N</td>
<td></td>
</tr>
</tbody>
</table>

Indicative timings

1. \( T_{\text{SMPL}} \) depends on SMP[2:0]
2. \( T_{\text{SAR}} \) depends on RES[2:0]

27.4.18 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP=1 and injected conversions ongoing by setting JADSTP=1.

Stopping conversions will reset the ongoing ADC operation. Then the ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC would re-start a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.
Note: In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

**Figure 163. Stopping ongoing regular conversions**

<table>
<thead>
<tr>
<th>ADC state</th>
<th>Trigger</th>
<th>Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDY</td>
<td>Sample Ch(N-1)</td>
<td>Convert Ch(N-1)</td>
</tr>
<tr>
<td>JADSTART</td>
<td>Cleared by SW</td>
<td>Cleared by HW</td>
</tr>
<tr>
<td>ADSTART</td>
<td>Cleared by SW</td>
<td>Cleared by HW</td>
</tr>
<tr>
<td>ADSTP</td>
<td>Cleared by HW</td>
<td>Cleared by HW</td>
</tr>
<tr>
<td>ADC_DR</td>
<td>Data N-2</td>
<td>Data N-1</td>
</tr>
</tbody>
</table>

**Figure 164. Stopping ongoing regular and injected conversions**

<table>
<thead>
<tr>
<th>ADC state</th>
<th>Regular trigger</th>
<th>Injected trigger</th>
<th>Regular trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDY</td>
<td>Sample Ch(N-1)</td>
<td>Sample Ch(M)</td>
<td>Sample Ch(M)</td>
</tr>
<tr>
<td>JADSTART</td>
<td>Set by S/W</td>
<td>Cleared by HW</td>
<td>Cleared by S/W</td>
</tr>
<tr>
<td>JADSTP</td>
<td>Set by S/W</td>
<td>Cleared by H/W</td>
<td>Cleared by H/W</td>
</tr>
<tr>
<td>ADC_JDR</td>
<td>DATA M-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADSTART</td>
<td>Set by S/W</td>
<td>Cleared by H/W</td>
<td>Cleared by S/W</td>
</tr>
<tr>
<td>ADSTP</td>
<td>REGULAR CONVERSIONS ongoing</td>
<td>Cleared by HW</td>
<td></td>
</tr>
<tr>
<td>ADC_DR</td>
<td>DATA N-2</td>
<td>DATA N-1</td>
<td></td>
</tr>
</tbody>
</table>
27.4.19 **Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)**

A conversion or a sequence of conversions can be triggered either by software or by an external event (e.g. timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

When the Injected Queue is enabled (bit JQDIS=0), injected software triggers are not possible.

The regular trigger selection is effective once software has set bit ADSTART=1 and the injected trigger selection is effective once software has set bit JADSTART=1.

Any hardware triggers which occur while a conversion is ongoing are ignored.
- If bit ADSTART=0, any regular hardware triggers which occur are ignored.
- If bit JADSTART=0, any injected hardware triggers which occur are ignored.

Table 188 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

<table>
<thead>
<tr>
<th>EXTEN[1:0]</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Hardware Trigger detection disabled, software trigger detection enabled</td>
</tr>
<tr>
<td>01</td>
<td>Hardware Trigger with detection on the rising edge</td>
</tr>
<tr>
<td>10</td>
<td>Hardware Trigger with detection on the falling edge</td>
</tr>
<tr>
<td>11</td>
<td>Hardware Trigger with detection on both the rising and falling edges</td>
</tr>
</tbody>
</table>

**Note:** The polarity of the regular trigger cannot be changed on-the-fly.

Table 189. Configuring the trigger polarity for injected external triggers

<table>
<thead>
<tr>
<th>JEXTEN[1:0]</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>- If JQDIS=1 (Queue disabled): Hardware trigger detection disabled, software trigger detection enabled</td>
</tr>
<tr>
<td></td>
<td>- If JQDIS=0 (Queue enabled): Hardware and software trigger detection disabled</td>
</tr>
<tr>
<td>01</td>
<td>Hardware Trigger with detection on the rising edge</td>
</tr>
<tr>
<td>10</td>
<td>Hardware Trigger with detection on the falling edge</td>
</tr>
<tr>
<td>11</td>
<td>Hardware Trigger with detection on both the rising and falling edges</td>
</tr>
</tbody>
</table>

**Note:** The polarity of the injected trigger can be anticipated and changed on-the-fly when the queue is enabled (JQDIS=0). Refer to Section 27.4.22: Queue of context for injected conversions.

The EXTSEL[4:0] and JEXTSEL[4:0] control bits select which out of 21 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.
Note: The regular trigger selection cannot be changed on-the-fly. The injected trigger selection can be anticipated and changed on-the-fly. Refer to Section 27.4.22: Queue of context for injected conversions on page 959.

Each ADC master shares the same input triggers with its ADC slave as described in Figure 165.

Figure 165. Triggers are shared between ADC master and ADC slave

Table 190 and Table 191 give all the possible external triggers of the three ADCs for regular and injected conversion.

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Type</th>
<th>EXTSEL[4:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>adc_ext_trg0</td>
<td>tim1_oc1</td>
<td>Internal signal from on-chip timers</td>
<td>00000</td>
</tr>
<tr>
<td>adc_ext_trg1</td>
<td>tim1_oc2</td>
<td>Internal signal from on-chip timers</td>
<td>00001</td>
</tr>
<tr>
<td>adc_ext_trg2</td>
<td>tim1_oc3</td>
<td>Internal signal from on-chip timers</td>
<td>00010</td>
</tr>
<tr>
<td>adc_ext_trg3</td>
<td>tim2_oc2</td>
<td>Internal signal from on-chip timers</td>
<td>00011</td>
</tr>
<tr>
<td>adc_ext_trg4</td>
<td>tim3_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>00100</td>
</tr>
<tr>
<td>adc_ext_trg5</td>
<td>tim4_oc4</td>
<td>Internal signal from on-chip timers</td>
<td>00101</td>
</tr>
<tr>
<td>adc_ext_trg6</td>
<td>exti11</td>
<td>External pin</td>
<td>00110</td>
</tr>
<tr>
<td>adc_ext_trg7</td>
<td>tim8_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>00111</td>
</tr>
<tr>
<td>adc_ext_trg8</td>
<td>tim8_trgo2</td>
<td>Internal signal from on-chip timers</td>
<td>01000</td>
</tr>
<tr>
<td>adc_ext_trg9</td>
<td>tim1_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>01001</td>
</tr>
</tbody>
</table>
### Table 190. ADC1 and ADC2 - External triggers for regular channels (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Type</th>
<th>EXTSEL[4:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>adc_ext_trg10</td>
<td>tim1_trgo2</td>
<td>Internal signal from on-chip timers</td>
<td>01010</td>
</tr>
<tr>
<td>adc_ext_trg11</td>
<td>tim2_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>01011</td>
</tr>
<tr>
<td>adc_ext_trg12</td>
<td>tim4_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>01100</td>
</tr>
<tr>
<td>adc_ext_trg13</td>
<td>tim6_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>01101</td>
</tr>
<tr>
<td>adc_ext_trg14</td>
<td>tim15_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>01110</td>
</tr>
<tr>
<td>adc_ext_trg15</td>
<td>tim3_oc4</td>
<td>Internal signal from on-chip timers</td>
<td>01111</td>
</tr>
<tr>
<td>adc_ext_trg16</td>
<td>Reserved</td>
<td>-</td>
<td>10000</td>
</tr>
<tr>
<td>adc_ext_trg17</td>
<td>Reserved</td>
<td>-</td>
<td>10001</td>
</tr>
<tr>
<td>adc_ext_trg18</td>
<td>lptim1_out</td>
<td>Internal signal from on-chip timers</td>
<td>10010</td>
</tr>
<tr>
<td>adc_ext_trg19</td>
<td>lptim2_out</td>
<td>Internal signal from on-chip timers</td>
<td>10011</td>
</tr>
<tr>
<td>adc_ext_trg20</td>
<td>lptim3_out</td>
<td>Internal signal from on-chip timers</td>
<td>10100</td>
</tr>
<tr>
<td>adc_ext_trg21</td>
<td>Reserved</td>
<td>-</td>
<td>10101</td>
</tr>
<tr>
<td>adc_ext_trg22</td>
<td>Reserved</td>
<td>-</td>
<td>10110</td>
</tr>
<tr>
<td>adc_ext_trg23</td>
<td>Reserved</td>
<td>-</td>
<td>10111</td>
</tr>
<tr>
<td>adc_ext_trg24</td>
<td>Reserved</td>
<td>-</td>
<td>11000</td>
</tr>
<tr>
<td>adc_ext_trg25</td>
<td>Reserved</td>
<td>-</td>
<td>11001</td>
</tr>
<tr>
<td>adc_ext_trg26</td>
<td>Reserved</td>
<td>-</td>
<td>11010</td>
</tr>
<tr>
<td>adc_ext_trg27</td>
<td>Reserved</td>
<td>-</td>
<td>11011</td>
</tr>
<tr>
<td>adc_ext_trg28</td>
<td>Reserved</td>
<td>-</td>
<td>11100</td>
</tr>
<tr>
<td>adc_ext_trg29</td>
<td>Reserved</td>
<td>-</td>
<td>11101</td>
</tr>
<tr>
<td>adc_ext_trg30</td>
<td>Reserved</td>
<td>-</td>
<td>11110</td>
</tr>
<tr>
<td>adc_ext_trg31</td>
<td>Reserved</td>
<td>-</td>
<td>11111</td>
</tr>
</tbody>
</table>

### Table 191. ADC1 and ADC2 - External triggers for injected channels

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Type</th>
<th>JEXTSEL[4:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>adc_jext_trg0</td>
<td>tim1_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>00000</td>
</tr>
<tr>
<td>adc_jext_trg1</td>
<td>tim1_oc4</td>
<td>Internal signal from on-chip timers</td>
<td>00001</td>
</tr>
<tr>
<td>adc_jext_trg2</td>
<td>tim2_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>00010</td>
</tr>
<tr>
<td>adc_jext_trg3</td>
<td>tim2_oc1</td>
<td>Internal signal from on-chip timers</td>
<td>00011</td>
</tr>
<tr>
<td>adc_jext_trg4</td>
<td>tim3_oc4</td>
<td>Internal signal from on-chip timers</td>
<td>00100</td>
</tr>
<tr>
<td>adc_jext_trg5</td>
<td>tim4_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>00101</td>
</tr>
<tr>
<td>adc_jext_trg6</td>
<td>exti15</td>
<td>External pin</td>
<td>00110</td>
</tr>
<tr>
<td>adc_jext_trg7</td>
<td>tim8_oc4</td>
<td>Internal signal from on-chip timers</td>
<td>00111</td>
</tr>
<tr>
<td>adc_jext_trg8</td>
<td>tim1_trgo2</td>
<td>Internal signal from on-chip timers</td>
<td>01000</td>
</tr>
<tr>
<td>adc_jext_trg9</td>
<td>tim8_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>01001</td>
</tr>
</tbody>
</table>
27.4.20 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.

2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Type</th>
<th>JEXTSEL[4:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>adc_jext_trg10</td>
<td>tim8_trgo2</td>
<td>Internal signal from on-chip timers</td>
<td>01010</td>
</tr>
<tr>
<td>adc_jext_trg11</td>
<td>tim3_oc3</td>
<td>Internal signal from on-chip timers</td>
<td>01011</td>
</tr>
<tr>
<td>adc_jext_trg12</td>
<td>tim3_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>01100</td>
</tr>
<tr>
<td>adc_jext_trg13</td>
<td>tim3_oc1</td>
<td>Internal signal from on-chip timers</td>
<td>01101</td>
</tr>
<tr>
<td>adc_jext_trg14</td>
<td>tim6_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>01110</td>
</tr>
<tr>
<td>adc_jext_trg15</td>
<td>tim15_trgo</td>
<td>Internal signal from on-chip timers</td>
<td>01111</td>
</tr>
<tr>
<td>adc_jext_trg16</td>
<td>Reserved</td>
<td>-</td>
<td>10000</td>
</tr>
<tr>
<td>adc_jext_trg17</td>
<td>Reserved</td>
<td>-</td>
<td>10001</td>
</tr>
<tr>
<td>adc_jext_trg18</td>
<td>lptim1_out</td>
<td>Internal signal from on-chip timers</td>
<td>10010</td>
</tr>
<tr>
<td>adc_jext_trg19</td>
<td>lptim2_out</td>
<td>Internal signal from on-chip timers</td>
<td>10011</td>
</tr>
<tr>
<td>adc_jext_trg20</td>
<td>lptim3_out</td>
<td>Internal signal from on-chip timers</td>
<td>10100</td>
</tr>
<tr>
<td>adc_jext_trg21</td>
<td>Reserved</td>
<td>-</td>
<td>10101</td>
</tr>
<tr>
<td>adc_jext_trg22</td>
<td>Reserved</td>
<td>-</td>
<td>10110</td>
</tr>
<tr>
<td>adc_jext_trg23</td>
<td>Reserved</td>
<td>-</td>
<td>10111</td>
</tr>
<tr>
<td>adc_jext_trg24</td>
<td>Reserved</td>
<td>-</td>
<td>11000</td>
</tr>
<tr>
<td>adc_jext_trg25</td>
<td>Reserved</td>
<td>-</td>
<td>11001</td>
</tr>
<tr>
<td>adc_jext_trg26</td>
<td>Reserved</td>
<td>-</td>
<td>11010</td>
</tr>
<tr>
<td>adc_jext_trg27</td>
<td>Reserved</td>
<td>-</td>
<td>11011</td>
</tr>
<tr>
<td>adc_jext_trg28</td>
<td>Reserved</td>
<td>-</td>
<td>11100</td>
</tr>
<tr>
<td>adc_jext_trg29</td>
<td>Reserved</td>
<td>-</td>
<td>11101</td>
</tr>
<tr>
<td>adc_jext_trg30</td>
<td>Reserved</td>
<td>-</td>
<td>11110</td>
</tr>
<tr>
<td>adc_jext_trg31</td>
<td>Reserved</td>
<td>-</td>
<td>11111</td>
</tr>
</tbody>
</table>
reset and the injected channel sequence switches are launched (all the injected
channels are converted once).

3. Then, the regular conversion of the regular group of channels is resumed from the last
interrupted regular conversion.

4. If a regular event occurs during an injected conversion, the injected conversion is not
interrupted but the regular sequence is executed at the end of the injected sequence.
*Figure 166* shows the corresponding timing diagram.

*Note:* When using triggered injection, one must ensure that the interval between trigger events is
longer than the injection sequence. For instance, if the sequence length is 20 ADC clock
cycles (that is two conversions with a sampling time of 1.5 clock periods), the minimum
interval between triggers must be 21 ADC clock cycles.

**Auto-injection mode**

If the JAUTO bit in the ADC_CFGR register is set, then the channels in the injected group
are automatically converted after the regular group of channels. This can be used to convert
a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR
registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular
conversions, followed by injected conversions (JADSTART must be kept cleared). Setting
the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected
channels are continuously converted.

*Note:* It is not possible to use both the auto-injected and discontinuous modes simultaneously.

When the DMA is used for exporting regular sequencer’s data in JAUTO mode, it is
necessary to program it in circular mode (CIRC bit set in DMA_CCRx register). If the CIRC
bit is reset (single-shot mode), the JAUTO sequence will be stopped upon DMA Transfer
Complete event.
27.4.21 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR register.

It is used to convert a short sequence (sub-group) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQR registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN=1, n=3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - ...

- DISCEN=0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - all the next trigger events will relaunch the complete sequence.
Note: When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN=1, CONT=1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where ‘n’ is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:
• JDISCEN=1, channels to be converted = 1, 2, 3
  – 1st trigger: channel 1 converted (a JEOC event is generated)
  – 2nd trigger: channel 2 converted (a JEOC event is generated)
  – 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  – ...

Note: When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

27.4.22 Queue of context for injected conversions

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions. JQDIS bit of ADC_CFGR register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled.

This context consists of:
• Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL[4:0] in ADC_JSQR register)
• Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADC_JSQR register)
All the parameters of the context are defined into a single register ADC_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The ADC_JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADC_CFGR:
  - If JQM=0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence will be served according to the last active context.
  - If JQM=1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and hardware triggers are disabled. Therefore, any further hardware injected triggers are ignored until the software re-writes a new injected context into JSQR register.
- Reading JSQR register returns the current JSQR context which is active at that moment. When the JSQR context is empty, JSQR is read as 0x0000.
- The Queue is flushed when stopping injected conversions by setting JADSTP=1 or when disabling the ADC by setting ADDIS=1:
  - If JQM=0, the Queue is maintained with the last active context.
  - If JQM=1, the Queue becomes empty and triggers are ignored.

**Note:** When configured in discontinuous mode (bit JDISCEN=1), only the last trigger of the injected sequence changes the context and consumes the Queue. The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):

- 1st trigger, discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out
- 2nd trigger, disc. Sequence 1: 2nd conversion.
- 3rd trigger, discontinuous. Sequence 1: 3rd conversion.
- 4th trigger, discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.
- 5th trigger, discontinuous. Sequence 2: 2nd conversion.
- 6th trigger, discontinuous. Sequence 2: 3rd conversion.

**Note:** When queue of context enabled (bit JQDIS=0), only hardware trigger can be used.
Behavior when changing the trigger or sequence context

The Figure 167 and Figure 168 show the behavior of the context Queue when changing the sequence or the triggers.

Figure 167. Example of JSQR queue of context (sequence change)

![Diagram showing behavior in sequence change context]

1. Parameters:
P1: sequence of 3 conversions, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 4 conversions, hardware trigger 1

Figure 168. Example of JSQR queue of context (trigger change)

![Diagram showing behavior in trigger change context]

1. Parameters:
P1: sequence of 2 conversions, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 2
P3: sequence of 4 conversions, hardware trigger 1
Queue of context: Behavior when a queue overflow occurs

The Figure 169 and Figure 170 show the behavior of the context Queue if an overflow occurs before or during a conversion.

Figure 169. Example of JSQR queue of context with overflow before conversion

1. Parameters:
P1: sequence of 2 conversions, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 2
P3: sequence of 3 conversions, hardware trigger 1
P4: sequence of 4 conversions, hardware trigger 1

Figure 170. Example of JSQR queue of context with overflow during conversion

1. Parameters:
P1: sequence of 2 conversions, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 2
P3: sequence of 3 conversions, hardware trigger 1
P4: sequence of 4 conversions, hardware trigger 1
It is recommended to manage the queue overflows as described below:

- After each P context write into JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

**Queue of context: Behavior when the queue becomes empty**

*Figure 171 and Figure 172* show the behavior of the context Queue when the Queue becomes empty in both cases JQM=0 or 1.

**Figure 171. Example of JSQR queue of context with empty queue (case JQM=0)**

![Queue Diagram](image)

1. Parameters:
   - P1: sequence of 1 conversion, hardware trigger 1
   - P2: sequence of 1 conversion, hardware trigger 1
   - P3: sequence of 1 conversion, hardware trigger 1

**Note:** When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.
Figure 172. Example of JSQR queue of context with empty queue (case JQM=1)

1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 173. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion.

1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1
Figure 174. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion and a new trigger occurs.

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Figure 175. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs outside an ongoing conversion

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1
Figure 176. Flushing JSQR queue of context by setting JADSTP=1 (JQM=1)

1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 177. Flushing JSQR queue of context by setting ADDIS=1 (JQM=0)

1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1
Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when JQM bit is reset:

1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

   Queue of context: Starting the ADC with an empty queue

   5. Write a dummy JSQR with JEXTEN not equal to 0 (otherwise triggering a software conversion)
   6. Set JADSTART
   7. Set JADSTP
   8. Wait until JADSTART is reset
   9. Set JADSTART.

   Disabling the queue

   It is possible to disable the queue by setting bit JQDIS=1 into the ADC_CFGR register.

Queue of context: Programming of the register ADC_JSQR

When the injected conversion queue of context is enabled (JQDIS=0), the ADC_JSQR must be programmed at one register write access. As JL[1:0] register define the number of the injected sequence, corresponding JSQ1 to JSQ4 must be written at same time. If ADC_JSQR is reprogrammed before the injected conversion start, reprogrammed data is put on the queue. When queue of context is empty, ADC_JSQR read back as 0x0000. Register access should not use the ‘read modify write’ sequence.

When ADC_JSQR is programmed when already 2 contexts are queued, it will raise JQOVF flag and generate the interrupt.
27.4.23 Programmable resolution (RES) - fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution. The resolution can be configured to be either 16, 14, 12, 10, 8 bits by programming the control bits RES[1:0]. Figure 183, Figure 184, Figure 185 and Figure 186 show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to Table 192.

![Figure 183](image1)
![Figure 184](image2)
![Figure 185](image3)
![Figure 186](image4)

Table 192. \(T_{SAR}\) timings depending on resolution

<table>
<thead>
<tr>
<th>RES [2:0]</th>
<th>(T_{SAR}) (ADC clock cycles)</th>
<th>(T_{SAR}) (ns) at (F_{adc_ker_ck}=24) MHz</th>
<th>(T_{adc_ker_ck}) (ADC clock cycles) (with Sampling Time=1.5 ADC clock cycles)</th>
<th>(T_{adc_ker_ck}) (ns) at (F_{adc_ker_ck}=24) MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 bits</td>
<td>8.5 ADC clock cycles</td>
<td>354.2</td>
<td>10 ADC clock cycles</td>
<td>416.7</td>
</tr>
<tr>
<td>14 bits</td>
<td>7.5 ADC clock cycles</td>
<td>312.5</td>
<td>9 ADC clock cycles</td>
<td>375</td>
</tr>
<tr>
<td>12 bits</td>
<td>6.5 ADC clock cycles</td>
<td>270.8</td>
<td>8 ADC clock cycles</td>
<td>333.3</td>
</tr>
<tr>
<td>10 bits</td>
<td>5.5 ADC clock cycles</td>
<td>229.2</td>
<td>7 ADC clock cycles</td>
<td>291.7</td>
</tr>
<tr>
<td>8 bits</td>
<td>4.5 ADC clock cycles</td>
<td>187.5</td>
<td>6 ADC clock cycles</td>
<td>250.0</td>
</tr>
</tbody>
</table>

27.4.24 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADC_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADC_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

27.4.25 End of conversion sequence (EOS, JEOS)

The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADC_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.
27.4.26  Timing diagrams example (single/continuous modes, hardware/software triggers)

**Figure 179. Single conversions of a sequence, software trigger**

1. EXTEN=0x0, CONT=0
2. Channels selected = 1, 9, 10, 17; AUTDLY=0.

**Figure 180. Continuous conversion of a sequence, software trigger**

1. EXTEN=0x0, CONT=1
2. Channels selected = 1, 9, 10, 17; AUTDLY=0.
27.4.27 Data management

Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE)

Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADC_DR data register which is 32 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADC_JDRy data register which is 32 bits wide.

The OVSS[3:0] and LSHIFT[3:0] bitfields in the ADC_CFGR2 register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in Figure 183, Figure 184, Figure 185 and Figure 186.

Note: The data can be re-aligned in normal and in oversampling mode.
Offset

An offset y (y=1, 2, 3, 4) can be applied to a channel by programming a value different from 0 in OFFSETy[25:0] bitfield into ADC_OFRy register. The channel to which the offset will be applied is programmed into the bits OFFSETy_CH[4:0] of ADC_OFRy register. In this case, the converted value is decreased by the user-defined offset written in the bits OFFSETy[25:0]. The result may be a negative value so the read data is signed and the SEXT bit represents the extended sign value.

The offset value should be lower than the max conversion value (ex. 16bit mode, offset value max is 0xFFFF).

The offset correction is also supported in oversampling mode. For the oversampling mode, offset is subtracted before OVSS right shift applied.

*Table 193* describes how the comparison is performed for all the possible resolutions for analog watchdog 1, 2, 3.

### Table 193. Offset computation versus data resolution

<table>
<thead>
<tr>
<th>Resolution (bits RES[2:0])</th>
<th>Subtraction between raw converted data and offset:</th>
<th>Result</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 bits</td>
<td>DATA[15:0], OFFSET[25:0]</td>
<td>signed 27-bit data</td>
<td>-</td>
</tr>
<tr>
<td>14 bits</td>
<td>DATA[15:2], OFFSET[25:0]</td>
<td>signed 27-bit data</td>
<td>The user must configure OFFSET[1:0] to 00</td>
</tr>
<tr>
<td>12 bits</td>
<td>DATA[15:4], OFFSET[25:0]</td>
<td>signed 27-bit data</td>
<td>The user must configure OFFSET[3:0] to 0000</td>
</tr>
<tr>
<td>10 bits</td>
<td>DATA[15:6], OFFSET[25:0]</td>
<td>signed 27-bit data</td>
<td>The user must configure OFFSET[5:0] to 000000</td>
</tr>
<tr>
<td>8 bits</td>
<td>DATA[15:8], OFFSET[25:0]</td>
<td>signed 27-bit data</td>
<td>The user must configure OFFSET[7:0] to 00000000</td>
</tr>
</tbody>
</table>

When reading data from ADC_DR (regular channel) or from ADC_JDRy (injected channel, y=1,2,3,4), the offset compensation is disabled when ADC_OFRy [25:0] bitfield is reset. Otherwise, the offset for ADC_OFRy[30:26] channel is enabled.

*Figure 183, Figure 184, Figure 185 and Figure 186* show alignments for signed and unsigned data together with corresponding OVSS and LSHIFT values.
### Figure 183. Right alignment (offset disabled, unsigned value)

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>23</th>
<th>15</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit data</td>
<td>0000</td>
<td>D15..D0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-bit data</td>
<td>0000</td>
<td>0</td>
<td>D11..D0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-bit data</td>
<td>0000</td>
<td>00</td>
<td>D7..D0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-bit data OSR=1024</td>
<td>00</td>
<td>0</td>
<td>D25..D0</td>
<td>OVSS = 0000</td>
<td></td>
</tr>
<tr>
<td>16-bit data OSR=1024</td>
<td>0000</td>
<td>D15..D0</td>
<td>OVSS = 1011</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Figure 184. Right alignment (offset enabled, signed value)

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>23</th>
<th>15</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit data</td>
<td>SEXT</td>
<td>D15..D0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-bit data</td>
<td>SEXT</td>
<td>D14..D0</td>
<td>SSATE = 1 OR RSHIFT1..4 = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-bit data</td>
<td>SEXT</td>
<td>D11..D0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-bit data</td>
<td>SEXT</td>
<td>D7..D0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-bit data</td>
<td>SEXT</td>
<td>D6..D0</td>
<td>SSATE = 1 OR RSHIFT1..4 = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-bit data OSR=1024</td>
<td>SEXT</td>
<td>D25..D0</td>
<td>OVSS = 0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
16-bit and 8-bit signed format management: RSHIFTx, SSATE

The offset correction sign-extends the data format, resulting in an unsigned 16-bit conversion being extended to 17-bit signed format, for instance.

Three options are offered for formatting 8-bit and 16-bit conversion results.

![Figure 185. Left alignment (offset disabled, unsigned value)](MSv41026V1)

<table>
<thead>
<tr>
<th>Format</th>
<th>LSHIFT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-bit data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-bit data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSR=1024</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Figure 186. Left alignment (offset enabled, signed value)](MSv41027V2)

<table>
<thead>
<tr>
<th>Format</th>
<th>LSHIFT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-bit data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-bit data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSR=1024</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For each offset correction channel 1 to 4, a RSHIFT1..4 bit in the ADC_CFGR2 register allows to have the result right-shifted 1-bit and have it fitting a standard 8 or 16-bit format.

Another option is to have the result saturated to the 16-bit and 8-bit signed formats, for the following cases only: RES[2:0] = 000 (16-bit format) and RES[2:0] = 111 (8-bit format).

This mode is enabled with the SSATE bit in the ADC_OFRy register.

The table below summarizes the 3 available use case for 16-bit format.

**Table 194. 16-bit data formats**

<table>
<thead>
<tr>
<th>SSATE</th>
<th>RSHIFTx</th>
<th>Format</th>
<th>Data range (offset = 0x8000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Sign-extended 17-bit significant data SEXT[31:16] DATA[15:0]</td>
<td>0x00007FFF - 0xFFFF8000</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Sign-extended right-shifted 16-bit significant data SEXT[31:15] DATA[14:0]</td>
<td>0x3FFF - 0xC000</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Sign-extended saturated 16-bit significant data SEXT[31:15] DATA[14:0]</td>
<td>7FFF - 0x8000</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Reserved</td>
<td>-</td>
</tr>
</tbody>
</table>

Numerical examples are given in Table 195 with 3 different offset values.

**Table 195. Numerical examples for 16-bit format (bold indicates saturation)**

<table>
<thead>
<tr>
<th>Raw conversion result</th>
<th>Offset value</th>
<th>Result SSATE = 0 RSHIFT = 0</th>
<th>Result SSATE = 0 RSHIFT = 1</th>
<th>Result SSATE = 1 RSHIFT = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFFFF</td>
<td>0x8000</td>
<td>0x0000 7FFF</td>
<td>3FFF</td>
<td>7FFF</td>
</tr>
<tr>
<td>0x8000</td>
<td>0x8000</td>
<td>0x0000 0000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0000</td>
<td>0xFFFF 8000</td>
<td>0xFFFF</td>
<td>C000</td>
<td>8000</td>
</tr>
<tr>
<td>0xFFFF</td>
<td>0x8020</td>
<td>0x0000 7DF</td>
<td>3FEF</td>
<td>7DFF</td>
</tr>
<tr>
<td>0x8000</td>
<td>0xFFFF FFE0</td>
<td>0xFFFF</td>
<td>FFE0</td>
<td>FFE0</td>
</tr>
<tr>
<td>0x0000</td>
<td>0xFFFF 7FE0</td>
<td>0xFFFF</td>
<td>BFF0</td>
<td>8000</td>
</tr>
<tr>
<td>0xFFFF</td>
<td>0x7FF0</td>
<td>0xFFFF 800F</td>
<td>4007</td>
<td>7FFF</td>
</tr>
<tr>
<td>0x8000</td>
<td>0x8000 0010</td>
<td>0x0000</td>
<td>8</td>
<td>0010</td>
</tr>
<tr>
<td>0x0000</td>
<td>0xFFFF 8010</td>
<td>0x0000 0008</td>
<td>C008</td>
<td>8010</td>
</tr>
</tbody>
</table>

When oversampling mode is active, the SSATE and RSHIFT1..4 bits are not supported.
ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) notifies of a buffer overrun event when the regular converted data has not been read (by the CPU or the DMA) before ADC_DR FIFO (8 stages) is overflowed. In this case, the OVR flag is set. An interrupt can be generated if OVRIE bit is set to 1.

When an overrun condition occurs, the ADC is still operating and can continue to convert unless the software decides to stop and reset the sequence by setting bit ADSTP=1.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0: The overrun event preserves the data register from being overwritten: the old data is maintained up to ADC_DR FIFO depth (8 data) and the new conversion is discarded and lost. If OVR remains at 1, any further conversions will occur but the result data will be also discarded.
- OVRMOD = 1: The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, any further conversions will operate normally and the ADC_DR register will always contain the latest converted data.

Figure 187. Example of overrun (OVR = 1 and OVRMOD = 1)

Note: There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversion without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD should be configured to 0 to manage overrun events or FIFO overflow as an error.
Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag should be ignored by the software. An overrun event will not prevent the ADC from continuing to convert and the ADC_DR register will always contain the latest conversion.

Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMNGT bit = 01 or 11 in the ADC_CFR register in single ADC mode or MDMA different from 0b00 in dual ADC mode), a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

Despite this, if an overrun occurs (OVR=1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to Section : ADC overrun (OVR, OVRMOD)).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMNGT of the ADC_CFR register in single ADC mode, or with bit DAMDF of the ADCx_CCR register in dual ADC mode:

- **DMA one shot mode (DMNGT bit = 01).**
  - This mode is suitable when the DMA is programmed to transfer a fixed number of data.

- **DMA circular mode (DMNGT bit = 11)**
  - This mode is suitable when programming the DMA in circular mode.

**DMA one shot mode (DMNGT=01)**

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when DMA_EOT interrupt occurs - refer to DMA paragraph) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted with partial result discarded.
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- Scan sequence is stopped and reset.
- The DMA is stopped.
DMA circular mode (DMNGT=11)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

DMA with FIFO

The output data register has 8 stage FIFO. Two different DMA requests are generated parallel. When a data is available, “SREQ single request” generated, when 4 data are available, “BREQ burst request” generated. DMA2 can be programmed either single transfer mode or incremental burst mode(4 beats), according to this mode, correct request line is selected by the DMA2. Please refer to the DMA2 chapter for further information.

27.4.28 Managing conversions using the DFSDM

The ADC conversion results can be transferred directly to the Digital Filter for Sigma Delta Modulators (DFSDM).

In this case, the DMNGT[1:0] bits must be set to 10.

The ADC transfers 16 least significant bits of the regular data register data to the DFSDM, which in turns will reset the EOC flag once the transfer is effective.

The data format must be 16-bit signed:

ADC_DR[31:16] = don’t care
ADC_DR[15] = sign
ADC_DR[14:0] = data

Any value above 16-bit signed format will be truncated.

27.4.29 Dynamic low-power features

Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When AUTDLY=1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADC_DR register has been read or if the EOC bit has been cleared (see Figure 188).
- For an injected conversion: when the JEOS bit has been cleared (see Figure 189).

This is a way to automatically adapt the speed of the ADC to the speed of the system which will read the data.

The delay is inserted after each regular conversion (whatever DISCEN=0 or 1) and after each sequence of injected conversions (whatever JDISCEN=0 or 1).

Note: There is no delay inserted between each conversions of the injected sequence, except after the last one.
During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

**Note:** This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to re-start a conversion: it is up to the software to read the data before launching a new conversion.

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see Figure 189).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see Figure 191).

The behavior is slightly different in auto-injected mode (JAUTO=1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see Figure 192).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO=1, CONT=1 and AUTDLY=1), follow the following procedure:

1. Wait until JEOS=1 (no more conversions are restarted)
2. Clear JEOS,
3. Set ADSTP=1
4. Read the regular data.

If this procedure is not respected, a new regular sequence can re-start if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.
Figure 188. AUTDLY=1, regular conversion in continuous mode, software trigger

1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, CHANNELS = 1, 2, 3
3. Injected configuration DISABLED

Figure 189. AUTDLY=1, regular HW conversions interrupted by injected conversions (DISCEN=0; JDISCEN=0)

1. AUTDLY=1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT=0, DISCEN =0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=0, CHANNELS = 5, 6
**Figure 190. AUTDLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1, JDISCEN=1)**

1. **AUTDLY=1**
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT=0, DISCEN=1, DISCNUM=1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=1, CHANNELS = 5, 6.
Figure 191. AUTDLY=1, regular continuous conversions interrupted by injected conversions

1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=0, CHANNELS = 5, 6

Figure 192. AUTDLY=1 in auto-injected mode (JAUTO=1)

1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2
3. Injected configuration: JAUTO=1, CHANNELS = 5, 6
27.4.30 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 193. Analog watchdog guarded area

**AWDx flag and interrupt**

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDyIE in the ADC_IER register (x=1,2,3).

AWDy (y=1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

**Description of analog watchdog 1**

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADC_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels\(^{(1)}\) remain within a configured voltage range (window).

*Table 196* shows how the ADC_CFGRy registers should be configured to enable the analog watchdog on one or more channels.

<table>
<thead>
<tr>
<th>Channels guarded by the analog watchdog</th>
<th>AWD1SGL bit</th>
<th>AWD1EN bit</th>
<th>JAWD1EN bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>All injected channels</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>All regular channels</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>All regular and injected channels</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Single(^{(1)}) injected channel</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Single(^{(1)}) regular channel</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Single(^{(1)}) regular or injected channel</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. Selected by the AWDyCH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.
These thresholds are programmed in bits HTR1[25:0] of the ADC_HTR1 register and LTR1[25:0] of the ADC_LTR1 register for the analog watchdog 1.

The threshold can be up to 26-bits (16-bit resolution with oversampling, OSVR[9:0]=1024).

When converting data with a resolution of less than 16 bits (according to bits RES[2:0]), the LSBs of the programmed thresholds must be kept cleared, the internal comparison being performed on the full 16-bit converted data (left aligned to the half-word boundary).

*Table 197* describes how the comparison is performed for all the possible resolutions for analog watchdog 1,2,3.

### Table 197. Analog watchdog 1,2,3 comparison

<table>
<thead>
<tr>
<th>Resolution (bit RES[2:0])</th>
<th>Analog watchdog comparison between:</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw converted data, left aligned (1)</td>
<td>Thresholds</td>
<td></td>
</tr>
<tr>
<td>16 bits</td>
<td>DATA[15:0]</td>
<td>LTR1[25:0] and HTR1[25:0]</td>
</tr>
<tr>
<td>14 bits</td>
<td>DATA[15:2],00</td>
<td>LTR1[25:0] and HTR1[25:0]</td>
</tr>
<tr>
<td>12 bits</td>
<td>DATA[15:4],0000</td>
<td>LTR1[25:0] and HTR1[25:0]</td>
</tr>
<tr>
<td>10 bits</td>
<td>DATA[15:6],00000</td>
<td>LTR1[25:0] and HTR1[25:0]</td>
</tr>
<tr>
<td>8 bits</td>
<td>DATA[15:8],000000</td>
<td>LTR1[25:0] and HTR1[25:0]</td>
</tr>
</tbody>
</table>

1. The watchdog comparison is performed on the raw converted data before any alignment calculation and before applying any offsets (the data which is compared is not signed).

**Description of analog watchdog 2 and 3**

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDCHy[19:0] (y=2,3).

The corresponding watchdog is enabled when any bit of AWDCHy[19:0] (y=2,3) is set.

The threshold can be up to 26-bits (16-bit resolution with oversampling, OSVR[9:0]=1024) and are programmed with the ADC_HTR2, ADC_LTR2, ADC_LTR3, and ADC_HTR3 registers.

When converting data with a resolution of less than 16 bits (according to bits RES[2:0]), the LSBs of the programmed thresholds must be kept cleared, the internal comparison being performed on the full 16-bit converted data (left aligned to the half-word boundary).

**ADCx_AWDy_OUT signal output generation**

Each analog watchdog is associated to an internal hardware signal ADCx_AWDy_OUT (x=ADC number, y=watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCx_AWDy_OUT signal as ETR.
ADCx_AWDy_OUT is activated when the associated analog watchdog is enabled:

- ADCx_AWDy_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCx_AWDy_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADCx_AWDy_OUT is also reset when disabling the ADC (when setting ADDIS=1).

Note: AWDx flag is set by hardware and reset by software: AWDy flag has no influence on the generation of ADCy_AWDx_OUT (ex: ADCy_AWDx_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).

Figure 194. ADCy_AWDx_OUT signal generation (on all regular channels)

![Diagram](MS31025V1)

Figure 195. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by SW)

![Diagram](MS31026V1)
27.4.31 Oversampler

The oversampling unit performs data preprocessing to offload the CPU. It is able to handle multiple conversions and average them into a single data with increased data width, up to 26-bit (16-bit values and OSVR[9:0] = 1024).

It provides a result with the following form, where N and M can be adjusted:

\[
\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)
\]

It allows to perform by hardware the following functions: averaging, data rate reduction, SNR improvement, basic filtering.
The oversampling ratio $N$ is defined using the $OSVR[9:0]$ bits in the ADC_CFGR2 register, and can range from $2x$ to $1024x$. The division coefficient $M$ consists of a right bit shift up to 10 bits, and is defined using the $OVSS[3:0]$ bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 26 bits ($1024 \times 16$-bit results), which can be left or right shifted. When right shifting is selected, it is rounded to the nearest value using the least significant bits left apart by the shifting, before being transferred into the ADC_DR data register.

The Table 198 gives a numerical example of the processing, from a raw 26-bit accumulated data to the final 16-bit result.

![Figure 198. 16-bit result oversampling with 10-bits right shift and rounding](MSv41028V2)

<table>
<thead>
<tr>
<th>16-bit data</th>
<th>OSVR=1024</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>OVSS[3:0]=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D25..D0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Right shifting and rounding</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16-bit data</th>
<th>OSVR=1024</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>OVSS[3:0]=1010</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D15..D0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16-bit data</th>
<th>OSVR=1024</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>OVSS[3:0]=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x3FFE258</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Right shifting and rounding</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16-bit data</th>
<th>OSVR=1024</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>OVSS[3:0]=1010</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0xFFF9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There are no changes for conversion timings in oversampled mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every $N$ conversions, with an equivalent delay equal to $N \times T_{CONV} = N \times (t_{SMPL} + t_{SAR})$. The flags are set as follow:

- the end of the sampling phase (EOSMP) is set after each sampling phase
- the end of conversion (EOC) occurs once every $N$ conversions, when the oversampled result is available
- the end of sequence (EOS) occurs once the sequence of oversampled data is completed (i.e. after $N \times$ sequence length conversions total)
Single ADC operating modes support when oversampling

In oversampling mode, most of the ADC operating modes are maintained:

- Single or continuous mode conversions
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[2:0] bits in ADC_CFGR register) are accumulated, truncated, rounded and shifted in the same way as 16-bit conversions are

**Note:** The alignment mode is not available when working with oversampled data. The data are always provided right-aligned.

Analog watchdog

The analog watchdog functionality is maintained (AWDSL and AWDEN bits), with the following difference:

- the RES[2:0] bits are ignored, comparison is always done on using the full 26-bit values HTRx[25:0] and LTRx[25:0]
- the comparison is performed on the oversampled accumulated value before shifting

**Note:** Care must be taken when using high shifting values, this will reduce the comparison range. For instance, if the oversampled result is shifted by 4 bits, thus yielding a 12-bit data right-aligned, the effective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HT[0:7] / LT[0:7], and HT[11:8] / LT[11:8] must be kept reset.

Triggered mode

The averager can also be used for basic filtering purpose. Although not a very powerful filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TROVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

The **Figure 199** below shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set, the content of the DISCEN bit is ignored and considered as 1.
Injected and regular sequencer management when oversampling

In oversampling mode, it is possible to have differentiated behavior for injected and regular sequencers. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

Oversampling regular channels only

The regular oversampling mode bit ROVSM defines how the regular oversampling sequence is resumed if it is interrupted by injected conversion:

- in continued mode, the accumulation re-starts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling will be completed whatever the injection frequency (providing at least one regular conversion can be completed between triggers);
- in resumed mode, the accumulation re-starts from 0 (previous conversions results are ignored). This mode allows to guarantee that all data used for oversampling were converted back-to-back within a single timeslot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be completed and the regular sequencer will be blocked.

The Figure 200 gives examples for a 4x oversampling ratio.
Figure 200. Regular oversampling modes (4x ratio)

Regular channels: \( \text{Ch(N)0, Ch(N)1, Ch(N)2, Ch(N)3, Ch(M)0, Ch(M)1, Ch(M)2, Ch(M)3} \)

Injected channels: \( \text{Ch(J), Ch(K)} \)

Oversampling stopped

Oversampling continued

Trigger

Abort

Figure 201. Regular and injected oversampling modes used simultaneously

Regular channels: \( \text{Ch(N)0, Ch(N)1, Ch(N)2, Ch(N)3, Ch(M)0, Ch(M)1, Ch(M)2, Ch(M)3} \)

Injected channels: \( \text{Ch(J), Ch(K)} \)

Oversampling aborted

Oversampling resumed

Trigger

Abort

Continued mode: \( \text{ROVSE} = 1, \text{JOVSE} = 0, \text{ROVSM} = 0, \text{TROVS} = X \)

Resumed mode: \( \text{ROVSE} = 1, \text{JOVSE} = 0, \text{ROVSM} = 1, \text{TROVS} = X \)

Oversampling Injected channels only

The Injected oversampling mode bit \( \text{JOVSE} \) enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and Injected channels

It is possible to have both \( \text{ROVSE} \) and \( \text{JOVSE} \) bits set. In this case, the regular oversampling mode is forced to resumed mode (\( \text{ROVSM} \) bit ignored), as represented on Figure 201 below.

Figure 201. Regular and injected oversampling modes used simultaneously

Regular channels: \( \text{Ch(N)0, Ch(N)1, Ch(N)2, Ch(N)3, Ch(M)0, Ch(M)1, Ch(M)2, Ch(M)3} \)

Injected channels: \( \text{Ch(J), Ch(K)} \)

Oversampling aborted

Oversampling resumed

Trigger

Abort

\( \text{ROVSE} = 1, \text{JOVSE} = 1, \text{ROVSM} = X, \text{TROVS} = 0 \)
Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected mode oversampling mode must be disabled, and the ROVSM bit is ignored (resumed mode is forced). The JOVSE bit must be reset. The behavior is represented on Figure 202 below.

![Figure 202. Triggered regular oversampling with injection](image)

Autoinjected mode

It is possible to oversample auto-injected sequences and have all conversions results stored in registers to save a DMA resource. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1, other combinations are not supported. The ROVSM bit is ignored in auto-injected mode. The Figure 203 below shows how the conversions are sequenced.

![Figure 203. Oversampling in auto-injected mode](image)

Dual ADC modes support when oversampling

It is possible to have oversampling enabled when working in dual ADC configuration, for the injected simultaneous mode and regular simultaneous mode. In this case, the two ADCs must be programmed with the very same settings (including oversampling).

All other dual ADC modes are not supported when either regular or injected oversampling is enabled (ROVSE = 1 or JOVSE = 1).
Combined modes summary

The Table 198 below summarizes all combinations, including modes not supported.

Table 198. Oversampler operating modes summary

<table>
<thead>
<tr>
<th>Regular Oversampling</th>
<th>Injected Oversampling</th>
<th>Oversampler mode</th>
<th>Triggered Regular mode</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROVSE</td>
<td>JOVSE</td>
<td>ROVSM</td>
<td>TROVS</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Regular continued mode</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Not supported</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Regular resumed mode</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Triggered regular resumed mode</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>Not supported</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Injected and regular resumed mode</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Not supported</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Injected oversampling</td>
</tr>
</tbody>
</table>

27.4.32 Dual ADC modes

In devices with two ADCs or more, dual ADC modes can be used (see Figure 204):
- ADC1 and ADC2 can be used together in dual mode (ADC1 is master)

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADCx master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADCx_CCR register.

Four possible modes are implemented:
- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:
- Injected simultaneous mode + Regular simultaneous mode
- Regular simultaneous mode + Alternate trigger mode
- Injected simultaneous mode + Interleaved mode

In dual ADC mode (when bits DUAL[4:0] in ADCx_CCR register are not equal to zero), the bits CONT, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JQM, JAUTO of the ADC_CFG register are shared between the master and slave ADC: the bits in the slave ADC are always equal to the corresponding bits of the master ADC.

To start a conversion in dual mode, the user must program the bits EXTEN, EXTSEL, JEXTEN, JEXTSEL of the master ADC only, to configure a software or hardware trigger,
and a regular or injected trigger. (the bits EXTEN[1:0] and JEXTEN[1:0] of the slave ADC are don’t care).

In regular simultaneous or interleaved modes: once the user sets bit ADSTART or bit ADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit ADSTART or bit ADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In injected simultaneous or alternate trigger modes: once the user sets bit JADSTART or bit JADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit JADSTART or bit JADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADCx_CDR). The status bits can be also read in parallel by reading the dual-mode status register (ADCx_CSR).
Figure 204. Dual ADC block diagram\(^{(1)}\)

1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.
2. The ADC common data register (ADCx_CDR) contains both the master and slave ADC regular converted data.
Injected simultaneous mode

This mode is selected by programming bits DUAL[4:0]=00101

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL[4:0] bits in the ADC_JSQR register).

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In simultaneous mode, one must convert sequences with the same length and inside a sequence, the N-th conversion in master ans slave must be configured with the same sampling time.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

- At the end of injected sequence of conversion event (JEOS) on the master ADC, the converted data is stored into the master ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- At the end of injected sequence of conversion event (JEOS) on the slave ADC, the converted data is stored into the slave ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- If the duration of the master injected sequence is equal to the duration of the slave injected one (like in Figure 205), it is possible for the software to enable only one of the two JEOS interrupt (ex: master JEOS) and read both converted data (from master ADC_JDRy and slave ADC_JDRy registers).

Figure 205. Injected simultaneous mode on 4 channels: dual ADC mode

If JDISCEN=1, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADC_DR) has been read. Any new regular trigger events occurring for the master ADC during the ongoing regular sequence and the associated delay phases are ignored. There is the same behavior for regular sequences occurring on the slave ADC.
Regular simultaneous mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00110.

This mode is performed on a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL[4:0] bits in the ADC_CFGR register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on master or on the slave) will abort the current simultaneous conversions, which are re-started once the injected conversion is completed.

**Note:** Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In regular simultaneous mode, one must convert sequences with the same length and inside a sequence, the N-th conversion in master ans slave must be configured with the same sampling time.

Software is notified by interrupts when it can read the data:
- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the slave ADC.
- If the duration of the master regular sequence is equal to the duration of the slave one (like in Figure 206), it is possible for the software to enable only one of the two EOC interrupt (ex: master EOC) and read both converted data from the Common Data register (ADCx_CDR).

It is also possible to read the regular data using the DMA. Two methods are possible:
- Using two DMA channels (one for the master and one for the slave). In this case bits DAMDF[1:0] must be kept cleared:
  - Configure the DMA master ADC channel to read ADC_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  - Configure the DMA slave ADC channel to read ADC_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Configuring Dual ADC mode data format DAMDF[1:0] bits, which leaves one DMA channel free for other uses:
  - Configure DAMDF[1:0]=0b10 or 0b11 (depending on resolution).
  - A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADCx_CDR)
  - A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADCx_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADCx_CCR register.
  - both EOC flags are cleared when the DMA reads the ADCx_CCR register.

**Note:** When DAMDF[1:0]=0b10 or 0b11, the user must program the same number of conversions in the master’s sequence as in the slave’s sequence. Otherwise, the remaining conversions will not generate a DMA request.
Figure 206. Regular simultaneous mode on 16 channels: dual ADC mode

If DISCEN=1 then each “n” simultaneous conversions of the regular sequence require a regular trigger event to occur ("n" is defined by DISCNUM).

This mode can be combined with AUTDLY mode:

- Once a simultaneous conversion of the sequence has ended, the next conversion in the sequence is started only if the common data register, ADCx_CDR (or the regular data register of the master ADC) has been read (delay phase).
- Once a simultaneous regular sequence of conversions has ended, a new regular trigger event is accepted only if the common data register (ADCx_CDR) has been read (delay phase). Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

It is possible to use the DMA to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multi-DMA mode is used: bits DAMDF must be set to 0b10 or 0b11.

When regular simultaneous mode is combined with AUTDLY mode, it is mandatory for the user to ensure that:

- The number of conversions in the master’s sequence is equal to the number of conversions in the slave’s.
- For each simultaneous conversions of the sequence, the length of the conversion of the slave ADC is inferior to the length of the conversion of the master ADC. Note that the length of the sequence depends on the number of channels to convert and the sampling time and the resolution of each channels.

**Note:** This combination of regular simultaneous mode and AUTDLY mode is restricted to the use case when only regular channels are programmed: it is forbidden to program injected channels in this combined mode.

**Interleaved mode with independent injected**

This mode is selected by programming bits DUAL[4:0] = 00111.

This mode can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately.
- The slave ADC starts after a delay of several-ADC clock cycles after the sampling phase of the master ADC has complete.

The minimum delay which separates 2 conversions in interleaved mode is configured in the DELAY bits in the ADCx_CCR register. This delay starts to count after the end of the sampling phase of the master conversion. This way, an ADC cannot start a conversion if the
complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

If the CONT bit is set on both master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

The software is notified by interrupts when it can read the data at the end of each conversion event (EOC) on the slave ADC. A slave and master EOC interrupts are generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave/master ADC.

Note: It is possible to enable only the EOC interrupt of the slave and read the common data register (ADCx_CDR). But in this case, the user must ensure that the duration of the conversions are compatible to ensure that inside the sequence, a master conversion is always followed by a slave conversion before a new master conversion restarts. It is recommended to use the MDMA mode.

It is also possible to have the regular data transferred by DMA. In this case, individual DMA requests on each ADC cannot be used and it is mandatory to use the MDMA mode, as following:

- Configure DAMDF[1:0]=0b10 or 0b11 (depending on resolution).
- A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADCx_CDR).
- A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADCx_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADCx_CCR register.
- Both EOC flags are cleared when the DMA reads the ADCx_CCR register.

Figure 207. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode
If DISCEN=1, each “n” simultaneous conversions (“n” is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In this mode, injected conversions are supported. When injection is done (either on master or on slave), both the master and the slave regular conversions are aborted and the sequence is re-started from the master (see Figure 209 below).

**Alternate trigger mode**

This mode is selected by programming bits DUAL[4:0] = 01001.

This mode can be started only on an injected group. The source of external trigger comes from the injected group multiplexer of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN must not be 0x0.
Injected discontinuous mode disabled (JDISCEN=0 for both ADC)

1. When the 1st trigger occurs, all injected master ADC channels in the group are converted.
2. When the 2nd trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on.

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts by converting the injected channels of the master ADC in the group.

Figure 210. Alternate trigger: injected group of each ADC

Note: Regular conversions can be enabled on one or all ADCs. In this case the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between 2 trigger events must be greater than or equal to 1 ADC clock period. The minimum time interval between 2 trigger events that start conversions on the same ADC is the same as in the single ADC mode.
Injected discontinuous mode enabled (JDISCEN=1 for both ADC)

If the injected discontinuous mode is enabled for both master and slave ADCs:

- When the 1st trigger occurs, the first injected channel of the master ADC is converted.
- When the 2nd trigger occurs, the first injected channel of the slave ADC is converted.
- And so on.

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversions.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts.

Figure 211. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode

Combined regular/injected simultaneous mode

This mode is selected by programming bits DUAL[4:0] = 00001.

It is possible to interrupt the simultaneous conversion of a regular group to start the simultaneous conversion of an injected group.

Note: The sequences must be converted with the same length, the N-th conversion in master and slave mode must be configured with the same sampling time inside a given sequence, or the interval between triggers has to be longer than the long conversion time of the 2 sequences. If the above conditions are not respected, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Combined regular simultaneous + alternate trigger mode

This mode is selected by programming bits DUAL[4:0]=00010.

It is possible to interrupt the simultaneous conversion of a regular group to start the alternate trigger conversion of an injected group. Figure 212 shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion is immediately started after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected
conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

Note: The sequences must be converted with the same length, the N-th conversion in master and slave mode must be configured with the same sampling time inside a given sequence, or the interval between triggers has to be longer than the long conversion time of the 2 sequences. If the above conditions are not respected, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Figure 212. Alternate + regular simultaneous

If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. Figure 213 shows the behavior in this case (note that the 6th trigger is ignored because the associated alternate conversion is not complete).

Figure 213. Case of trigger occurring during injected conversion

Combined injected simultaneous plus interleaved

This mode is selected by programming bits DUAL[4:0]=00011

It is possible to interrupt an interleaved conversion with a simultaneous injected event.

In this case the interleaved conversion is interrupted immediately and the simultaneous injected conversion starts. At the end of the injected sequence the interleaved conversion is resumed. When the interleaved regular conversion resumes, the first regular conversion which is performed is always the master’s one. Figure 214, Figure 215 and Figure 216 show the behavior using an example.

Caution: In this mode, it is mandatory to use the Common Data Register to read the regular data with a single read access. On the contrary, master-slave data coherency is not guaranteed.
Figure 214. Interleaved single channel CH0 with injected sequence CH11, CH12

Figure 215. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12
- case 1: Master interrupted first

Figure 216. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12
- case 2: Slave interrupted first
DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to Figure 217: DMA Requests in regular simultaneous mode when DAMDF=0b00).

**Figure 217. DMA Requests in regular simultaneous mode when DAMDF=0b00**

In simultaneous regular and interleaved modes, it is also possible to save one DMA channel and transfer both data using a single DMA channel. For this DAMDF bits must be configured in the ADCx_CCR register:

- **DAMDF=0b10, 32-bit format**: A single DMA request is generated alternatively when either the master or slave EOC events have occurred. At that time, the data items are alternatively available in the ADCx_CDR2 32-bit register. This mode is used in interleaved mode and in regular simultaneous mode when resolution is above 16-bit. **Example:**
  
  Interleaved dual mode: a DMA request is generated each time a new 32-bit data is available:
  
  1st DMA request: ADCx_CDR2[31:0] = MST_ADC_DR[31:0]
  2nd DMA request: ADCx_CDR2[31:0] = SLV_ADC_DR[31:0]

- **DAMDF=0b10, 16-bit format**: A single DMA request is generated each time both master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADCx_CDR contains the two half-words representing two ADC-
converted data items. The slave ADC data take the upper half-word and the master ADC data take the lower half-word.

This mode is used in interleaved mode and in regular simultaneous mode when resolution is ranging from 10 to 16-bit. Any value above 16-bit in the master or the slave converter will be truncated to the least 16 significant bits.

**Example:**

Interleaved dual mode: a DMA request is generated each time 2 data items are available:

1st DMA request: ADCx_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]  
2nd DMA request: ADCx_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

**Figure 218. DMA requests in regular simultaneous mode when DAMDF=0b10**

**Figure 219. DMA requests in interleaved mode when DAMDF=0b10**
Note: When using Multi ADC mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion is available.

- **DAMDF=0b11**: This mode is similar to the DAMDF=0b10. The only differences are that on each DMA request (two data items are available), two bytes representing two ADC converted data items are transferred as a half-word.
  
  This mode is used in interleaved and regular simultaneous mode when the result is 8-bit. A new DMA request is issued when 4 new 8-bit values are available.

**Example:**

Interleaved dual mode: a DMA request is generated each time 4 data items are available (t0, t1,... are corresponding to the consecutive sampling instants)

1st DMA request:

\[
\begin{align*}
\text{ADCx CDR}[7:0] &= \text{MST ADC DR}[7:0]_{t0} \\
\text{ADCx CDR}[15:8] &= \text{SLV ADC DR}[7:0]_{t0} \\
\text{ADCx CDR}[23:16] &= \text{MST ADC DR}[7:0]_{t1} \\
\text{ADCx CDR}[31:24] &= \text{SLV ADC DR}[7:0]_{t1}
\end{align*}
\]

2nd DMA request:

\[
\begin{align*}
\text{ADCx CDR}[7:0] &= \text{MST ADC DR}[7:0]_{t2} \\
\text{ADCx CDR}[15:8] &= \text{SLV ADC DR}[7:0]_{t2} \\
\text{ADCx CDR}[23:16] &= \text{MST ADC DR}[7:0]_{t3} \\
\text{ADCx CDR}[31:24] &= \text{SLV ADC DR}[7:0]_{t3}
\end{align*}
\]

**Overrun detection**

In dual ADC mode (when DUAL[4:0] is not equal to b00000), if an overrun is detected on one of the ADCs, the DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the DAMDF configuration).

It may happen that the EOC bit corresponding to one ADC remains set because the data register of this ADC contains valid data.

**DMA one shot mode/ DMA circular mode when Multi ADC mode is selected**

When DAMDF mode is selected (0b10 or 0b11), bit DMNGT[1:0]=0b10 in the master ADC’s ADCx CCR register must also be configured to select between DMA one shot mode and circular mode, as explained in section *Section : Managing conversions using the DMA*.

**Stopping the conversions in dual ADC modes**

The user must set the control bits ADSTP/JADSTP of the master ADC to stop the conversions of both ADC in dual ADC mode. The other ADSTP control bit of the slave ADC has no effect in dual ADC mode.

Once both ADC are effectively stopped, the bits ADSTART/JADSTART of the master and slave ADCs are both cleared by hardware.
**DFSDM mode in dual ADC mode interleaved mode**

In dual ADC interleaved modes, the ADC conversion results can be transferred directly to the Digital Filter for Sigma Delta Modulators (DFSDM).

This mode is enabled by setting the bits DMNGT[1:0] = 0b10 in the master ADC’s ADC_CFGR register.

The ADC transfers alternatively the 16 least significant bits of the regular data register from the master and the slave converter to a single channel of the DFSDM.

The data format must be 16-bit signed:

- ADC_DR[31:16] = 0x0000
- ADC_DR[15] = sign
- ADC_DR[14:0] = data

Any value above 16-bit signed format in any converter will be truncated.

**DFSDM mode in dual ADC simultaneous mode**

The dual mode is not required to use DFSDM in dual ADC simultaneous mode since conversion data will be treated by each individual channel. Single mode with same trigger source results in simultaneous conversion with DFSDM interface.

### 27.4.33 Temperature sensor

The temperature sensor can measure the junction temperature \(T_J\) of the device in the –40 to 125 °C temperature range.

The temperature sensor is internally connected to an ADC internal channel which is used to convert the sensor’s output voltage to a digital value (refer to **Table ADC interconnection** in **Section 27.4.2: ADC pins and internal signals**). The sampling time for the temperature sensor’s analog pin must be greater than the stabilization time specified in the product datasheet.

When not in use, the sensor can be put in power-down mode.

*Figure 220* shows the block diagram of the temperature sensor.

![Figure 220. Temperature sensor channel block diagram](image)

**Note:** The TSEN bit must be set to enable the conversion of the corresponding ADC internal channel (temperature sensor, \(V_{SENSE}\)).
Reading the temperature

To use the sensor:

1. Select the ADC input channels (with the appropriate sampling time).
2. Program with the appropriate sampling time (refer to electrical characteristics section of the device datasheet).
3. Set the TSEN bit in the ADCx_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting $V_{SENSE}$ data in the ADC data register.
6. Calculate the actual temperature using the following formula:

   \[
   \text{Temperature (in } ^\circ\text{C}) = \frac{110 \text{ } ^\circ\text{C} - 30 \text{ } ^\circ\text{C}}{\text{TS_CAL2} - \text{TS_CAL1}} \times (\text{TS_DATA} - \text{TS_CAL1}) + 30 \text{ } ^\circ\text{C}
   \]

   Where:
   - TS_CAL2 is the temperature sensor calibration value acquired at 110°C
   - TS_CAL1 is the temperature sensor calibration value acquired at 30°C
   - TS_DATA is the actual temperature sensor output value converted by ADC
   
   Refer to the device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

   Note: The sensor has a startup time after waking from power-down mode before it can output $V_{SENSE}$ at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and SENSEEN bits should be set at the same time.

27.4.34 $V_{BAT}$ supply monitoring

The VBATEN bit of the ADCx_CCR register is used to switch to the battery voltage. As the $V_{BAT}$ voltage could be higher than $V_{DDA}$, to ensure the correct operation of the ADC, the VBAT pin is internally connected to a bridge divider by 4. This bridge is automatically enabled when VBATEN is set, to connect $V_{BAT}/4$ to an ADC input channel (refer to Table ADC interconnection in Section 27.4.2: ADC pins and internal signals).

As a consequence, the converted digital value is one fourth of the $V_{BAT}$ voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the $V_{BAT}/4$ voltage.

Figure 221 shows the block diagram of the $V_{BAT}$ sensing feature.
27.4.35 Monitoring the internal voltage reference

The internal voltage reference can be monitored to have a reference point for evaluating the ADC $V_{REF+}$ voltage level.

The internal voltage reference is internally connected to an ADC input channel (refer to Table ADC interconnection in Section 27.4.2: ADC pins and internal signals).

The sampling time for this channel must be greater than the stabilization time specified in the product datasheet.

*Figure 221* shows the block diagram of the $V_{REF^{int}}$ sensing feature.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{vrefint_block_diagram.png}
\caption{V_{REF^{int}} channel block diagram}
\end{figure}

Note: The VREF\text{EN} bit of the ADCx CCR register must be configured to enable the conversion of the corresponding ADC internal channel ($V_{REF^{int}}$).
Calculating the actual $V_{DDA}$ voltage using the internal reference voltage

The power supply voltage applied to the device may be subject to variations or not precisely known. When $V_{DDA}$ is connected to $V_{REF+}$, it is possible to compute the actual $V_{DDA}$ voltage using the embedded internal reference voltage ($V_{REFINT}$). $V_{REFINT}$ and its calibration data acquired by the ADC during the manufacturing process at $V_{DDA} = 3.3 \text{ V}$ can be used to evaluate the actual $V_{DDA}$ voltage level.

The following formula gives the actual $V_{DDA}$ voltage supplying the device:

$$V_{REF+} = 3.3 \text{ V} \times \frac{V_{REFINT}_{\text{CAL}}}{V_{REFINT}_{\text{DATA}}}$$

Where:
- $V_{REFINT}_{\text{CAL}}$ is the $V_{REFINT}$ calibration value
- $V_{REFINT}_{\text{DATA}}$ is the actual $V_{REFINT}$ output value converted by ADC

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between $V_{REF+}$ and the voltage applied on the converted channel.

For most applications $V_{DDA}$ value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from $V_{DDA}$:

$$V_{\text{CHANNELx}} = \frac{V_{REF+}}{\text{FULL\_SCALE}} \times \text{ADC\_DATA}$$

By replacing $V_{REF+}$ by the formula provided above, the absolute voltage value is given by the following formula:

$$V_{\text{CHANNELx}} = \frac{3.3 \text{ V} \times V_{REFINT}_{\text{CAL}} \times \text{ADC\_DATA}}{V_{REFINT}_{\text{DATA}} \times \text{FULL\_SCALE}}$$

For applications where $V_{DDA}$ is known and ADC converted values are right-aligned, the absolute voltage value can be obtained by using the following formula:

$$V_{\text{CHANNELx}} = \frac{V_{DDA}}{\text{FULL\_SCALE}} \times \text{ADC\_DATA}$$

Where:
- $V_{REFINT}_{\text{CAL}}$ is the $V_{REFINT}$ calibration value
- $\text{ADC\_DATA}$ is the value measured by the ADC on channel x (right-aligned)
- $V_{REFINT}_{\text{DATA}}$ is the actual $V_{REFINT}$ output value converted by the ADC
- $\text{FULL\_SCALE}$ is the maximum digital value of the ADC output. For example with 16-bit resolution, it will be $2^{16} - 1 = 65535$ or with 8-bit resolution, $2^8 - 1 = 255$

Note: If ADC measurements are done using an output format other than 16-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.
27.5 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (OVR flag)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.

Table 199. ADC interrupts per each ADC

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC ready</td>
<td>ADRDY</td>
<td>ADRDYIE</td>
</tr>
<tr>
<td>End of conversion of a regular group</td>
<td>EOC</td>
<td>EOCIE</td>
</tr>
<tr>
<td>End of sequence of conversions of a regular group</td>
<td>EOS</td>
<td>EOSIE</td>
</tr>
<tr>
<td>End of conversion of a injected group</td>
<td>JEOC</td>
<td>JEOCIE</td>
</tr>
<tr>
<td>End of sequence of conversions of an injected group</td>
<td>JEOS</td>
<td>JEOSIE</td>
</tr>
<tr>
<td>Analog watchdog 1 status bit is set</td>
<td>AWD1</td>
<td>AWD1IE</td>
</tr>
<tr>
<td>Analog watchdog 2 status bit is set</td>
<td>AWD2</td>
<td>AWD2IE</td>
</tr>
<tr>
<td>Analog watchdog 3 status bit is set</td>
<td>AWD3</td>
<td>AWD3IE</td>
</tr>
<tr>
<td>End of sampling phase</td>
<td>EOSMP</td>
<td>EOSMPIE</td>
</tr>
<tr>
<td>Overrun</td>
<td>OVR</td>
<td>OVRIE</td>
</tr>
<tr>
<td>Injected context queue overflows</td>
<td>JQOVF</td>
<td>JQOVFIE</td>
</tr>
</tbody>
</table>
27.6 **ADC registers (for each ADC)**

Refer to *Section 1.2 on page 97* for a list of abbreviations used in register descriptions.

27.6.1 **ADC interrupt and status register (ADC_ISR)**

Address offset: 0x00

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDORDY</td>
<td>JQOVF</td>
<td>AWD3</td>
<td>AWD2</td>
<td>AWD1</td>
<td>JEOS</td>
<td>JEOC</td>
<td>OVR</td>
<td>EOS</td>
<td>EOC</td>
<td>EOSMP</td>
<td>ADRDY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDY**: ADC LDO output voltage ready bit

This bit is set and cleared by hardware. It indicates that the ADC internal LDO output is ready and that the ADC can be enabled or calibrated.

0: ADC LDO voltage regulator disabled
1: ADC LDO voltage regulator enabled

*Note: Refer to Section 27.3: ADC implementation* for the availability of the LDO regulator.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to *Section 27.4.22: Queue of context for injected conversions* for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)
1: Injected context queue overflow has occurred

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADC_TR3 register. It is cleared by software writing 1 to it.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)
1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADC_TR2 register. It is cleared by software writing 1 to it.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)
1: Analog watchdog 2 event occurred
Bit 7  **AWD1**: Analog watchdog 1 flag
This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADC_TR1 register. It is cleared by software, writing 1 to it.
0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)
1: Analog watchdog 1 event occurred

Bit 6  **JEOS**: Injected channel end of sequence flag
This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.
0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)
1: Injected conversions complete

Bit 5  **JEOC**: Injected channel end of conversion flag
This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADC_JDRy register
0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)
1: Injected channel conversion complete

Bit 4  **OVR**: ADC overrun
This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed while the EOC flag was already set. It is cleared by software writing 1 to it.
0: No overrun occurred (or the flag event was already acknowledged and cleared by software)
1: Overrun has occurred

Bit 3  **EOS**: End of regular sequence flag
This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.
0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)
1: Regular Conversions sequence complete
Bit 2  **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register.

- 0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)
- 1: Regular channel conversion complete

Bit 1  **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.

- 0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)
- 1: End of sampling phase reached

Bit 0  **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN=1) and when the ADC reaches a state where it is ready to accept conversion requests.

- It is cleared by software writing 1 to it.
- 0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)
- 1: ADC is ready to start conversion
27.6.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x04
Reset value: 0x0000 0000

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

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable
This bit is set and cleared by software to enable/disable the Injected Context Queue Overflow interrupt.
0: Injected Context Queue Overflow interrupt disabled
1: Injected Context Queue Overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

*Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).*

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable
This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.
0: Analog watchdog 3 interrupt disabled
1: Analog watchdog 3 interrupt enabled

*Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).*

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable
This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.
0: Analog watchdog 2 interrupt disabled
1: Analog watchdog 2 interrupt enabled

*Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).*

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable
This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.
0: Analog watchdog 1 interrupt disabled
1: Analog watchdog 1 interrupt enabled

*Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).*

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable
This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.
0: JEOS interrupt disabled
1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

*Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).*
Bit 5  **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.
0: JEOC interrupt disabled.
1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

*Note: The software is allowed to write this bit only when JADSTART is cleared to 0 (no injected conversion is ongoing).*

Bit 4  **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.
0: Overrun interrupt disabled
1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

*Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).*

Bit 3  **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.
0: EOS interrupt disabled
1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

*Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).*

Bit 2  **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.
0: EOC interrupt disabled
1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

*Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).*

Bit 1  **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.
0: EOSMP interrupt disabled.
1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

*Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).*

Bit 0  **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.
0: ADRDY interrupt disabled
1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

*Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).*
### 27.6.3 ADC control register (ADC_CR)

Address offset: 0x08
Reset value: 0x2000 0000

![Table](attachment:image.png)

**Bit 31** ADCAL: ADC calibration
This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or differential inputs mode. It is cleared by hardware after calibration is complete.

- 0: Calibration complete
- 1: Write 1 to calibrate the ADC. Read at 1 means that a calibration in progress.

**Note:** The software is allowed to launch a calibration by setting ADCAL only when ADEN=0.
The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).

**Bit 30** ADCALDIF: Differential mode for calibration
This bit is set and cleared by software to configure the single-ended or differential inputs mode for the calibration.

- 0: Writing ADCAL will launch a calibration in Single-ended inputs Mode.
- 1: Writing ADCAL will launch a calibration in Differential inputs Mode.

**Note:** The software is allowed to write this bit only when the ADC is disabled and not calibrating (ADCAL=0, JADSTART=0, ADSTART=0, ADDIS=0 and ADEN=0).

**Bit 29** DEEPPWD: Deep-power-down enable
This bit is set and cleared by software to put the ADC in deep-power-down mode.

- 0: ADC not in deep-power down
- 1: ADC in deep-power-down (default reset state)

**Note:** The software is allowed to write this bit only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADDIS=0 and ADEN=0).

**Bit 28** ADVREGEN: ADC voltage regulator enable
This bits is set by software to enable the ADC voltage regulator. Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

- 0: ADC Voltage regulator disabled
- 1: ADC Voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to Section 27.4.6: ADC deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN).

The software can program this bitfield only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADDIS=0 and ADEN=0).
Bit 27 **LINCALRDYW6**: Linearity calibration ready Word 6

This control / status bit allows to read/write the 6th linearity calibration factor.

When the linearity calibration is complete, this bit is set. A bit clear will launch the transfer of the linearity factor 6 into the **ADC_CALFACT2[29:0]**. The bit will be reset by hardware when the **ADC_CALFACT2** register can be read (software must poll the bit until it is cleared).

When the LINCALRDYW6 bit is reset, a new linearity factor 6 value can be written into the LINCALFACT[29:0] of the ADC_CALFACT2 register. A bit set will launch the linearity factor 6 update and the bit will be effectively set by hardware once the update will be done (software must poll the bit until it is set to indicate the write is effective).

**Note:** ADC_CALFACT2[29:10] contains 0. ADC_CALFACT2[9:0] corresponds linearity correction factor bits[159:150]. The software is allowed to toggle this bit only if the LINCALRDYW5, LINCALRDYW4, LINCALRDYW3, LINCALRDYW2 and LINCALRDYW1 bits are left unchanged, see chapter 27.4.8: Calibration (ADCAL, ADCALDIF, ADCALLIN, ADC_CALFACT) for details.

The software is allowed to update the linearity calibration factor by writing LINCALRDYWx only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing)

Bit 26 **LINCALRDYW5**: Linearity calibration ready Word 5

Refer to LINCALRDYW6 description.

**Note:** ADC_CALFACT2[29:0] corresponds linearity correction factor bits[149:120]. The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW3, LINCALRDYW2 and LINCALRDYW1 bits are left unchanged.

Bit 25 **LINCALRDYW4**: Linearity calibration ready Word 4

Refer to LINCALRDYW6 description.

**Note:** ADC_CALFACT2[29:0] correspond linearity correction factor bits[119:90]. The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW3, LINCALRDYW2 and LINCALRDYW1 bits are left unchanged.

Bit 24 **LINCALRDYW3**: Linearity calibration ready Word 3

Refer to LINCALRDYW6 description.

**Note:** ADC_CALFACT2[29:0] corresponds linearity correction factor bits[89:60]. The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW4, LINCALRDYW3 and LINCALRDYW1 bits are left unchanged.

Bit 23 **LINCALRDYW2**: Linearity calibration ready Word 2

Refer to LINCALRDYW6 description.

**Note:** ADC_CALFACT2[29:0] corresponds linearity correction factor bits[59:30]. The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW4, LINCALRDYW3 and LINCALRDYW1 bits are left unchanged.

Bit 22 **LINCALRDYW1**: Linearity calibration ready Word 1

Refer to LINCALRDYW6 description.

**Note:** ADC_CALFACT2[29:0] corresponds linearity correction factor bits[29:0]. The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW4, LINCALRDYW3 and LINCALRDYW2 bits are left unchanged.

Bits 21:17  Reserved, must be kept at reset value.
**Bit 16 ADCALLIN**: Linearity calibration
This bit is set and cleared by software to enable the Linearity calibration.
0: Writing ADCAL will launch a calibration without the Linearity calibration.
1: Writing ADCAL will launch a calibration with the Linearity calibration.

*Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).*

**Bits 15:10** Reserved, must be kept at reset value.

**Bits 9:8** \(\text{BOOST}[1:0]\): Boost mode control
This bitfield is set and cleared by software to enable/disable the Boost mode.
00: used when ADC clock \(\leq 6.25\) MHz
01: used when 6.25 MHz < ADC clock frequency \(\leq 12.5\) MHz
10: used when 12.5 MHz < ADC clock \(\leq 25.0\) MHz
11: used when 25.0 MHz < ADC clock \(\leq 50.0\) MHz

*Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

When dual mode is enabled (bits DAMDF of ADCx_CCR register are not equal to zero), the BOOST bitfield of the slave ADC is no more writable and its content must be equal to the master ADC BOOST bitfield.*

**Bits 7:6** Reserved, must be kept at reset value.

**Bit 5 JADSTP**: ADC stop of injected conversion command
This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command).
It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be re-configured. The ADC is then ready to accept a new start of injected conversions (JADSTP command).
0: No ADC stop injected conversion command ongoing
1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

*Note: The software is allowed to set JADSTP only when JADSTART=1 and ADDIS=0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable the ADC).
In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)*

**Bit 4 ADSTP**: ADC stop of regular conversion command
This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command).
It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).
0: No ADC stop regular conversion command ongoing
1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

*Note: The software is allowed to set ADSTP only when ADSTART=1 and ADDIS=0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable the ADC).
In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).
In dual ADC regular simultaneous mode and interleaved mode, the bit ADSTP of the master ADC must be used to stop regular conversions. The other ADSTP bit is inactive.*
Bit 3  **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN, a conversion will start immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:
- in single conversion mode when software trigger is selected (JEXTSEL=0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.
- in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.
1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

*Note:* The software is allowed to set JADSTART only when ADEN=1 and ADDIS=0 (ADC is enabled and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO=1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)
Bit 2  **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN, a conversion will start immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:
- in single conversion mode (CONT=0, DISCEN=0) when software trigger is selected (EXTEN=0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.
- In discontinuous conversion mode (CONT=0, DISCEN=1), when the software trigger is selected (EXTEN=0x0): at the end of conversion (EOC) flag.
- in all other cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.
1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

*Note:* The software is allowed to set ADSTART only when ADEN=1 and ADDIS=0 (ADC is enabled and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO=1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 1  **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing
1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

*Note:* The software is allowed to set ADDIS only when ADEN=1 and both ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

Bit 0  **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC will be effectively ready to operate once the flag ADRDY has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)
1: Write 1 to enable the ADC.

*Note:* The software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)
27.6.4 ADC configuration register (ADC_CFGR)

Address offset: 0x0C
Reset value: 0x8000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>JQDIS</th>
<th>Injected Queue disable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>These bits are set and cleared by software to disable the Injected Queue mechanism:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Injected Queue enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Injected Queue disabled</td>
</tr>
<tr>
<td>Note:</td>
<td>The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no regular nor injected conversion is ongoing).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A set or reset of JQDIS bit causes the injected queue to be flushed and the JSQR register is cleared.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30:26</th>
<th>AWD1CH[4:0]</th>
<th>Analog watchdog 1 channel selection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00000: ADC analog input channel-0 monitored by AWD1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00001: ADC analog input channel-1 monitored by AWD1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>.....</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10010: ADC analog input channel-19 monitored by AWD1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>others: Reserved, must not be used</td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td>The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 25</th>
<th>JAUTO: Automatic injected group conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.</td>
</tr>
<tr>
<td></td>
<td>0: Automatic injected group conversion disabled</td>
</tr>
<tr>
<td></td>
<td>1: Automatic injected group conversion enabled</td>
</tr>
<tr>
<td>Note:</td>
<td>The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no regular nor injected conversion is ongoing).</td>
</tr>
<tr>
<td></td>
<td>When dual mode is enabled (DAMDF bits in ADCx_CCR register are not equal to zero), the bit JAUTO of the slave ADC is no more writable and its content is equal to the bit JAUTO of the master ADC.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 24</th>
<th>JAWD1EN: Analog watchdog 1 enable on injected channels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and cleared by software</td>
</tr>
<tr>
<td></td>
<td>0: Analog watchdog 1 disabled on injected channels</td>
</tr>
<tr>
<td></td>
<td>1: Analog watchdog 1 enabled on injected channels</td>
</tr>
<tr>
<td>Note:</td>
<td>The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).</td>
</tr>
</tbody>
</table>
Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels
This bit is set and cleared by software.
0: Analog watchdog 1 disabled on regular channels
1: Analog watchdog 1 enabled on regular channels

*Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).*

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels
This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels.
0: Analog watchdog 1 enabled on all channels
1: Analog watchdog 1 enabled on a single channel

*Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).*

Bit 21 **JQM**: JSQR queue mode
This bit is set and cleared by software.
It defines how an empty Queue is managed.
0: JSQR Mode 0: The Queue is never empty and maintains the last written configuration into JSQR.
1: JSQR Mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to Section 27.4.22: Queue of context for injected conversions for more information.

*Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).*

When dual mode is enabled (DAMDF bits in ADCx_CCR register are not equal to zero), the bit JQM of the slave ADC is no more writable and its content is equal to the bit JQM of the master ADC.

Bit 20 **JDISCEN**: Discontinuous mode on injected channels
This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.
0: Discontinuous mode on injected channels disabled
1: Discontinuous mode on injected channels enabled

*Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).*

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

When dual mode is enabled (bits DAMDF of ADCx_CCR register are not equal to zero), the bit JDISCEN of the slave ADC is no more writable and its content is equal to the bit JDISCEN of the master ADC.

Bits 19:17 **DISCNUM[2:0]**: Discontinuous mode channel count
These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.
000: 1 channel
001: 2 channels
... 111: 8 channels

*Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).*

When dual mode is enabled (DAMDF bits in ADCx_CCR register are not equal to zero), the bits DISCNUM[2:0] of the slave ADC are no more writable and their content is equal to the bits DISCNUM[2:0] of the master ADC.
Bit 16 **DISCEN**: Discontinuous mode for regular channels
This bit is set and cleared by software to enable/disable Discontinuous mode for regular channels.
0: Discontinuous mode for regular channels disabled
1: Discontinuous mode for regular channels enabled

**Note:** It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.
It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.
The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).
When dual mode is enabled (DAMDF bits in ADCx_CCR register are not equal to zero), the bit DISCEN of the slave ADC is no more writable and its content is equal to the bit DISCEN of the master ADC.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **AUTDLY**: Delayed conversion mode
This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode.
0: Auto-delayed conversion mode off
1: Auto-delayed conversion mode on

**Note:** The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).
When dual mode is enabled (DAMDF bits in ADCx_CCR register are not equal to zero), the bit AUTDLY of the slave ADC is no more writable and its content is equal to the bit AUTDLY of the master ADC.

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions
This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.
0: Single conversion mode
1: Continuous conversion mode

**Note:** It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.
The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).
When dual mode is enabled (DAMDF bits in ADCx_CCR register are not equal to zero), the bit CONT of the slave ADC is no more writable and its content is equal to the bit CONT of the master ADC.

Bit 12 **OVRMOD**: Overrun Mode
This bit is set and cleared by software and configure the way data overrun is managed.
0: ADC_DR register is preserved with the old data when an overrun is detected.
1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

**Note:** The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels
These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.
00: Hardware trigger detection disabled (conversions can be launched by software)
01: Hardware trigger detection on the rising edge
10: Hardware trigger detection on the falling edge
11: Hardware trigger detection on both the rising and falling edges

**Note:** The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).
Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

- 00000: Event 0
- 00001: Event 1
- 00010: Event 2
- 00011: Event 3
- 00100: Event 4
- 00101: Event 5
- 00110: Event 6
- 00111: Event 7
- ...  
- 11111: Event 31

*Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).*

Bits 4:2 **RES[2:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

- 000: 16 bits
- 001: 14 bits in legacy mode (not optimized power consumption)
- 010: 12 bits in legacy mode (not optimized power consumption)
- 101: 14 bits
- 110: 12 bits
- 011: 10 bits
- 111: 8 bits

Others: Reserved, must not be used.

*Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).*

Bits 1:0 **DMNGT[1:0]**: Data Management configuration

This bit is set and cleared by software to select how ADC interface output data are managed.

- 00: Regular conversion data stored in DR only
- 01: DMA One Shot Mode selected
- 10: DFSDM mode selected
- 11: DMA Circular Mode selected

*Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).*

In dual-ADC modes, this bit is not relevant and replaced by control bit DAMDF of the **ADCx_CCR** register.
# ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10</td>
<td>0x0000 0000</td>
<td>ADC configuration register 2 (ADC_CFGR2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSHIFT[3:0]</td>
<td>31 30 29 28</td>
<td>Left shift factor</td>
</tr>
<tr>
<td></td>
<td>27 26</td>
<td>This bitfield is set and cleared by software to define the left shifting</td>
</tr>
<tr>
<td></td>
<td>25 24</td>
<td>applied to the final result with or without oversampling.</td>
</tr>
<tr>
<td></td>
<td>23 22</td>
<td>0000: No left shift</td>
</tr>
<tr>
<td></td>
<td>21 20</td>
<td>0001: Shift left 1-bit</td>
</tr>
<tr>
<td></td>
<td>19 18</td>
<td>0010: Shift left 2-bits</td>
</tr>
<tr>
<td></td>
<td>17 16</td>
<td>0011: Shift left 3-bits</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0100: Shift left 4-bits</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>0101: Shift left 5-bits</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>0110: Shift left 6-bits</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>0111: Shift left 7-bits</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>1000: Shift left 8-bits</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>1001: Shift left 9-bits</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>1010: Shift left 10-bits</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1011: Shift left 11-bits</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>1100: Shift left 12-bits</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>1101: Shift left 13-bits</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1110: Shift left 14-bits</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1111: Shift left 15-bits</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSVR[9:0]</td>
<td>31 30</td>
<td>Oversampling ratio</td>
</tr>
<tr>
<td></td>
<td>29 28</td>
<td>This bitfield is set and cleared by software to define the oversampling</td>
</tr>
<tr>
<td></td>
<td>27 26</td>
<td>ratio.</td>
</tr>
<tr>
<td></td>
<td>25 24</td>
<td>0: 1x (no oversampling)</td>
</tr>
<tr>
<td></td>
<td>23 22</td>
<td>1: 2x</td>
</tr>
<tr>
<td></td>
<td>21 20</td>
<td>2: 3x</td>
</tr>
<tr>
<td></td>
<td>19 18</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>17 16</td>
<td>1023: 1024x</td>
</tr>
</tbody>
</table>

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>27 26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSHIFT4</td>
<td>25</td>
<td>Right-shift data after Offset 4 correction</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>Refer to RSHIFT1 description.</td>
</tr>
</tbody>
</table>

Bits 31:26 LSHIFT[3:0]: Left shift factor

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:16 OSVR[9:0]: Oversampling ratio

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bit 15 Reserved, must be kept at reset value.

Bit 14 RSHIFT4: Right-shift data after Offset 4 correction

Refer to RSHIFT1 description.
Bit 13 **RSHIFT3**: Right-shift data after Offset 3 correction
Refer to RSHIFT1 description

Bit 12 **RSHIFT2**: Right-shift data after Offset 2 correction
Refer to RSHIFT1 description

Bit 11 **RSHIFT1**: Right-shift data after Offset 1 correction
This bitfield is set and cleared by software to right-shift 1-bit data after offset1 correction. This bit can only be used for 8-bit and 16-bit data format (see Section: Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE) for details).
0: Right-shifting disabled
1: Data is right-shifted 1-bit.

Bit 10 **ROVSM**: Regular Oversampling mode
This bit is set and cleared by software to select the regular oversampling mode.
0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)
1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bit 9 **TROVS**: Triggered Regular Oversampling
This bit is set and cleared by software to enable triggered oversampling
0: All oversampled conversions for a channel are done consecutively following a trigger
1: Each oversampled conversion for a channel needs a new trigger

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: Oversampling right shift
This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.
0000: No right shift
0001: Shift right 1-bit
0010: Shift right 2-bits
0011: Shift right 3-bits
0100: Shift right 4-bits
0101: Shift right 5-bits
0110: Shift right 6-bits
0111: Shift right 7-bits
1000: Shift right 8-bits
1001: Shift right 9-bits
1010: Shift right 10-bits
1011: Shift right 11-bits
Others: Reserved, must not be used.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no conversion is ongoing).
Bits 4:2 Reserved, must be kept at reset value.

**Bit 1 JOVSE: Injected Oversampling Enable**

This bit is set and cleared by software to enable injected oversampling.

- 0: Injected Oversampling disabled
- 1: Injected Oversampling enabled

**Note:** The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

**Bit 0 ROVSE: Regular Oversampling Enable**

This bit is set and cleared by software to enable regular oversampling.

- 0: Regular Oversampling disabled
- 1: Regular Oversampling enabled

**Note:** The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

### 27.6.6 ADC sample time register 1 (ADC_SMPR1)

**Address offset:** 0x14

**Reset value:** 0x0000 0000

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

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

**Bits 29:0 SMP[9:0][2:0]: Channel x sampling time selection (x = 0 to 9)**

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

- 000: 1.5 ADC clock cycles
- 001: 2.5 ADC clock cycles
- 010: 8.5 ADC clock cycles
- 011: 16.5 ADC clock cycles
- 100: 32.5 ADC clock cycles
- 101: 64.5 ADC clock cycles
- 110: 387.5 ADC clock cycles
- 111: 810.5 ADC clock cycles

**Note:** The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).
27.6.7  ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x18
Reset value: 0x0000 0000

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

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

Bits 29:0  **SMP[19:10][2:0]**: Channel x sampling time selection (x = 10 to 19)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

- 000: 1.5 ADC clock cycles
- 001: 2.5 ADC clock cycles
- 010: 8.5 ADC clock cycles
- 011: 16.5 ADC clock cycles
- 100: 32.5 ADC clock cycles
- 101: 64.5 ADC clock cycles
- 110: 387.5 ADC clock cycles
- 111: 810.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).
### 27.6.8 ADC channel preselection register (ADC_PCSEL)

Address offset: 0x1C  
Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 PCSEL[19:0]: Channel x (VinP[i]) pre selection (x = 0 to 19)

These bits are written by software to pre select the input channel at IO instance to be converted.

- 0: Input Channel x (VinP x) is not pre selected for conversion, the ADC conversion result with this channel shows wrong result.
- 1: Input Channel x (VinP x) is pre selected for conversion

**Note:** The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

### 27.6.9 ADC watchdog threshold register 1 (ADC_LTR1)

Address offset: 0x20  
Reset value: 0x0000 0000

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 LTR1[25:0]: Analog watchdog 1 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 1.

Refer to Section 27.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRY, AWD_LTRY, AWDy)

**Note:** The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).
27.6.10    ADC watchdog threshold register 1 (ADC_HTR1)

Address offset: 0x24
Reset value: 0x03FF FFFF

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

Bits 31:26  Reserved, must be kept at reset value.

Bits 25:0  HTR1[25:0]: Analog watchdog 1 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog 1.
Refer to Section 27.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).
### 27.6.11 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x30  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:29</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>
| 28:24 | SQ4[4:0]: 4th conversion in regular sequence  
These bits are written by software with the channel number (0..19) assigned as the 4th in the regular conversion sequence. |
| 23    | Reserved, must be kept at reset value. |
| 22:18 | SQ3[4:0]: 3rd conversion in regular sequence  
These bits are written by software with the channel number (0..19) assigned as the 3rd in the regular conversion sequence.  
**Note:** The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing). |
| 17    | Reserved, must be kept at reset value. |
| 16:12 | SQ2[4:0]: 2nd conversion in regular sequence  
These bits are written by software with the channel number (0..19) assigned as the 2nd in the regular conversion sequence.  
**Note:** The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing). |
| 11    | Reserved, must be kept at reset value. |
| 10:6  | SQ1[4:0]: 1st conversion in regular sequence  
These bits are written by software with the channel number (0..19) assigned as the 1st in the regular conversion sequence.  
**Note:** The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing). |
| 5:4   | Reserved, must be kept at reset value. |
| 3:0   | L[3:0]: Regular channel sequence length  
These bits are written by software to define the total number of conversions in the regular channel conversion sequence.  
0000: 1 conversion  
0001: 2 conversions  
...  
1111: 16 conversions  
**Note:** The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing). |
## 27.6.12 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x34

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:29</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 28:24</td>
<td><strong>SQ9[4:0]</strong>: 9th conversion in regular sequence</td>
</tr>
<tr>
<td></td>
<td>These bits are written by software with the channel number (0..19) assigned as the 9th in the regular conversion sequence.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).</td>
</tr>
<tr>
<td>Bit 23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 22:18</td>
<td><strong>SQ8[4:0]</strong>: 8th conversion in regular sequence</td>
</tr>
<tr>
<td></td>
<td>These bits are written by software with the channel number (0..19) assigned as the 8th in the regular conversion sequence.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).</td>
</tr>
<tr>
<td>Bit 17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 16:12</td>
<td><strong>SQ7[4:0]</strong>: 7th conversion in regular sequence</td>
</tr>
<tr>
<td></td>
<td>These bits are written by software with the channel number (0..19) assigned as the 7th in the regular conversion sequence.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).</td>
</tr>
<tr>
<td>Bit 11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 10:6</td>
<td><strong>SQ6[4:0]</strong>: 6th conversion in regular sequence</td>
</tr>
<tr>
<td></td>
<td>These bits are written by software with the channel number (0..19) assigned as the 6th in the regular conversion sequence.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 4:0</td>
<td><strong>SQ5[4:0]</strong>: 5th conversion in regular sequence</td>
</tr>
<tr>
<td></td>
<td>These bits are written by software with the channel number (0..19) assigned as the 5th in the regular conversion sequence.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).</td>
</tr>
</tbody>
</table>
### 27.6.13 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x38  
Reset value: 0x0000 0000

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

- **Bits 31:29**: Reserved, must be kept at reset value.
- **Bits 28:24**: **SQ14[4:0]**: 14th conversion in regular sequence  
  These bits are written by software with the channel number (0..19) assigned as the 14th in the regular conversion sequence.  
  **Note**: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).
- **Bit 23**: Reserved, must be kept at reset value.
- **Bits 22:18**: **SQ13[4:0]**: 13th conversion in regular sequence  
  These bits are written by software with the channel number (0..19) assigned as the 13th in the regular conversion sequence.  
  **Note**: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).
- **Bit 17**: Reserved, must be kept at reset value.
- **Bits 16:12**: **SQ12[4:0]**: 12th conversion in regular sequence  
  These bits are written by software with the channel number (0..19) assigned as the 12th in the regular conversion sequence.  
  **Note**: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).
- **Bit 11**: Reserved, must be kept at reset value.
- **Bits 10:6**: **SQ11[4:0]**: 11th conversion in regular sequence  
  These bits are written by software with the channel number (0..19) assigned as the 11th in the regular conversion sequence.  
  **Note**: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).
- **Bit 5**: Reserved, must be kept at reset value.
- **Bits 4:0**: **SQ10[4:0]**: 10th conversion in regular sequence  
  These bits are written by software with the channel number (0..19) assigned as the 10th in the regular conversion sequence.  
  **Note**: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).
### 27.6.14 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x3C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:11</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 10:6</td>
<td><strong>SQ16[4:0]</strong>: 16th conversion in regular sequence</td>
</tr>
<tr>
<td></td>
<td>These bits are written by software with the channel number (0..19) assigned as the 16th in the regular conversion sequence.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: <em>The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).</em></td>
</tr>
<tr>
<td>Bit 5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 4:0</td>
<td><strong>SQ15[4:0]</strong>: 15th conversion in regular sequence</td>
</tr>
<tr>
<td></td>
<td>These bits are written by software with the channel number (0..19) assigned as the 15th in the regular conversion sequence.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: <em>The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).</em></td>
</tr>
</tbody>
</table>
27.6.15 ADC regular Data Register (ADC_DR)

Address offset: 0x40
Reset value: 0x0000 0000

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

**RDATA[31:16]**

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

**RDATA[15:0]**

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

Bits 31:0 **RDATA[31:0]**: Regular Data converted
These bits are read-only. They contain the conversion result from the last converted regular channel.
The data are left- or right-aligned as described in Section 27.4.27: Data management.
27.6.16 ADC injected sequence register (ADC_JSQR)

Address offset: 0x4C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address offset</th>
<th>Address offset</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x4C</td>
<td>0x4C</td>
<td>0x4C</td>
</tr>
</tbody>
</table>

| Bits 31:27 JSQ4[4:0]: 4th conversion in the injected sequence |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| These bits are written by software with the channel number (0..19) assigned as the 4th in the injected conversion sequence. |
| Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register). |
| Bit 26 Reserved, must be kept at reset value. |

| Bits 25:21 JSQ3[4:0]: 3rd conversion in the injected sequence |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| These bits are written by software with the channel number (0..19) assigned as the 3rd in the injected conversion sequence. |
| Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register). |
| Bit 20 Reserved, must be kept at reset value. |

| Bits 19:15 JSQ2[4:0]: 2nd conversion in the injected sequence |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| These bits are written by software with the channel number (0..19) assigned as the 2nd in the injected conversion sequence. |
| Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register). |
| Bit 14 Reserved, must be kept at reset value. |

| Bits 13:9 JSQ1[4:0]: 1st conversion in the injected sequence |
| --- | --- | --- | --- | --- | --- |
| These bits are written by software with the channel number (0..19) assigned as the 1st in the injected conversion sequence. |
| Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register). |
Bits 8:7  **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

- 00: If JQDIS=0 (queue enabled), Hardware and software trigger detection disabled and
- 01: Hardware trigger detection on the rising edge
- 10: Hardware trigger detection on the falling edge
- 11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing).

If JQM=1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to Section 27.4.22: Queue of context for injected conversions)

Bits 6:2  **JEXTSEL[4:0]**: External trigger selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

- 00000: Event 0
- 00001: Event 1
- 00010: Event 2
- 00011: Event 3
- 00100: Event 4
- 00101: Event 5
- 00110: Event 6
- 00111: Event 7
- ...
- 11111: Event 31:

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing).

Bits 1:0  **JL[1:0]**: Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

- 00: 1 conversion
- 01: 2 conversions
- 10: 3 conversions
- 11: 4 conversions

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing).
27.6.17 ADC injected channel y offset register (ADC_OFRy)

Address offset: 0x60 + 0x04 * (y-1), (y= 1 to 4)

Reset value: 0x0000 0000

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

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 30:26 OFFSETy_CH[4:0]: Channel selection for the Data offset y

These bits are written by software to define the channel to which the offset programmed into bits OFFSETy[25:0] will apply.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 25:0 OFFSETy[25:0]: Data offset y for the channel programmed into bits OFFSETy_CH[4:0]

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the bits OFFSETy_CH[4:0]. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

When OFFSETy[25:0] bitfield is reset, the offset compensation is disabled.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

If several offset (OFFSETy) point to the same channel, only the offset with the lowest x value is considered for the subtraction.

Ex: if OFFSET1_CH[4:0]=4 and OFFSET2_CH[4:0]=4, this is OFFSET1[25:0] which is subtracted when converting channel 4.
27.6.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x80 + 0x04 * (y-1), (y= 1 to 4)
Reset value: 0x0000 0000

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

Bits 31:0 JDATA[31:0]: Injected data
These bits are read-only. They contain the conversion result from injected channel y. The data are left-or right-aligned as described in Section 27.4.27: Data management.

27.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR)

Address offset: 0xA0
Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 AWD2CH[19:0]: Analog watchdog 2 channel selection
These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWD2CH[i] = 0: ADC analog input channel-i is not monitored by AWD2
AWD2CH[i] = 1: ADC analog input channel-i is monitored by AWD2
When AWD2CH[19:0] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWD2CH must be also selected into the SQRi or JSQRi registers. The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).
27.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0xA4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>WWD3CH[19:16]</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWD3CH[19:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWD3CH[i] = 0: ADC analog input channel-i is not monitored by AWD3
AWD3CH[i] = 1: ADC analog input channel-i is monitored by AWD3

When AWD3CH[19:0] = 000..0, the analog Watchdog 3 is disabled

**Note:** The channels selected by AWD3CH must be also selected into the SQRi or JSQRi registers.
The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

27.6.21 ADC watchdog lower threshold register 2 (ADC_LTR2)

Address offset: 0xB0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>LTR2[25:16]</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **LTR2[25:0]**: Analog watchdog 2 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 2.

Refer to Section 27.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy).

**Note:** The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).
**27.6.22 ADC watchdog higher threshold register 2 (ADC_HTR2)**

Address offset: 0xB4
Reset value: 0x03FF FFFF

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **HTR2[25:0]**: Analog watchdog 2 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 2.

Refer to Section 27.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRY, AWD_LTRY, AWDy).

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

**27.6.23 ADC watchdog lower threshold register 3 (ADC_LTR3)**

Address offset: 0xB8
Reset value: 0x0000 0000

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **LTR3[25:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 3.

Refer to Section 27.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRY, AWD_LTRY, AWDy).

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).
27.6.24 ADC watchdog higher threshold register 3 (ADC_HTR3)

Address offset: 0xBC
Reset value: 0x03FF FFFF

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 HTR3[25:0]: Analog watchdog 3 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog 3.

Refer to Section 27.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRY, AWD_LTRY, AWDy)

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

27.6.25 ADC differential mode selection register (ADC_DIFSEL)

Address offset: 0xC0
Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 DIFSEL[19:0]: Differential mode for channels 19 to 0
These bits are set and cleared by software. They allow to select if a channel is configured as single ended or differential mode.

DIFSEL[i] = 0: ADC analog input channel-i is configured in single ended mode
DIFSEL[i] = 1: ADC analog input channel-i is configured in differential mode

Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).
27.6.26 ADC calibration factors register (ADC_CALFACT)

Address offset: 0xC4
Reset value: 0x0000 0000

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

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

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

Bits 26:16 **CALFACT_D[10:0]**: Calibration Factors in differential mode
These bits are written by hardware or by software.
Once a differential inputs calibration is complete, they are updated by hardware with the calibration factors.
Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it will then be applied once a new differential conversion is launched.

**Note**: The software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

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

Bits 10:0 **CALFACT_S[10:0]**: Calibration Factors in Single-Ended mode
These bits are written by hardware or by software.
Once a single-ended inputs calibration is complete, they are updated by hardware with the calibration factors.
Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it will then be applied once a new single-ended conversion is launched.

**Note**: The software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

27.6.27 ADC calibration factor register 2 (ADC_CALFACT2)

Address offset: 0xC8
Reset value: 0x0000 0000

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

| LINCALFACT[15:0] | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
27.7 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

27.7.1 ADC xcommon status register (ADCx_CSR) (x=1/2)

Address offset: 0x00
Reset value: 0x0000 0000

The address offset is relative to the master ADC base address + 0x300.

This register provides an image of the status bits of the different ADCs. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADC_ISR register.

ADC1 and ADC2 are controlled by the same interface.

```
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td>26</td>
<td>JQOVF_SLV</td>
</tr>
<tr>
<td>25</td>
<td>AWD3_SLV</td>
</tr>
<tr>
<td>24</td>
<td>AWD2_SLV</td>
</tr>
<tr>
<td>23</td>
<td>AWD1_SLV</td>
</tr>
<tr>
<td>22</td>
<td>JEOS_SLV</td>
</tr>
<tr>
<td>21</td>
<td>JEOSC_SLV</td>
</tr>
<tr>
<td>20</td>
<td>OVR_SLV</td>
</tr>
<tr>
<td>19</td>
<td>EOS_SLV</td>
</tr>
<tr>
<td>18</td>
<td>EOC_SLV</td>
</tr>
<tr>
<td>17</td>
<td>EOSMP_SLV</td>
</tr>
<tr>
<td>16</td>
<td>ADRDY_SLV</td>
</tr>
</tbody>
</table>
```

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

Bit 26 **JQOVF_SLV**: Injected Context Queue Overflow flag of the slave ADC
This bit is a copy of the JQOVF bit in the corresponding ADCx+1_ISR register.

Bit 25 **AWD3_SLV**: Analog watchdog 3 flag of the slave ADC
This bit is a copy of the AWD3 bit in the corresponding ADCx+1_ISR register.

Bit 24 **AWD2_SLV**: Analog watchdog 2 flag of the slave ADC
This bit is a copy of the AWD2 bit in the corresponding ADCx+1_ISR register.

Bit 23 **AWD1_SLV**: Analog watchdog 1 flag of the slave ADC
This bit is a copy of the AWD1 bit in the corresponding ADCx+1_ISR register.
Bit 22 **JEOS_SLV**: End of injected sequence flag of the slave ADC  
This bit is a copy of the JEOS bit in the corresponding ADCx+1_ISR register.

Bit 21 **JEOC_SLV**: End of injected conversion flag of the slave ADC  
This bit is a copy of the JEOC bit in the corresponding ADCx+1_ISR register.

Bit 20 **OVR_SLV**: Overrun flag of the slave ADC  
This bit is a copy of the OVR bit in the corresponding ADCx+1_ISR register.

Bit 19 **EOS_SLV**: End of regular sequence flag of the slave ADC  
This bit is a copy of the EOS bit in the corresponding ADCx+1_ISR register.

Bit 18 **EOC_SLV**: End of regular conversion of the slave ADC  
This bit is a copy of the EOC bit in the corresponding ADCx+1_ISR register.

Bit 17 **EOSMP_SLV**: End of Sampling phase flag of the slave ADC  
This bit is a copy of the EOSMP2 bit in the corresponding ADCx+1_ISR register.

Bit 16 **ADRDY_SLV**: Slave ADC ready  
This bit is a copy of the ADRDY bit in the corresponding ADCx+1_ISR register.

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

Bit 10 **JQOVF_MST**: Injected Context Queue Overflow flag of the master ADC  
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 9 **AWD3_MST**: Analog watchdog 3 flag of the master ADC  
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 **AWD2_MST**: Analog watchdog 2 flag of the master ADC  
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 **AWD1_MST**: Analog watchdog 1 flag of the master ADC  
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 **JEOS_MST**: End of injected sequence flag of the master ADC  
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 **JEOC_MST**: End of injected conversion flag of the master ADC  
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 **OVR_MST**: Overrun flag of the master ADC  
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3 **EOS_MST**: End of regular sequence flag of the master ADC  
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 **EOC_MST**: End of regular conversion of the master ADC  
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 **EOSMP_MST**: End of Sampling phase flag of the master ADC  
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 **ADRDY_MST**: Master ADC ready  
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.
### 27.7.2 ADC x common control register (ADCx_CCR) (x=1/2)

Address offset: 0x08  
Reset value: 0x0000 0000

The address offset is relative to the master ADC base address + 0x300.

ADC1 and ADC2 are controlled by the same interface.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VBATEN</td>
<td>TSEN</td>
<td>VREFEN</td>
<td>PRESC[3:0]</td>
<td>CKMODE[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 31:25 Reserved, must be kept at reset value.

**Bit 24 VBATEN**: VBAT enable  
This bit is set and cleared by software to control VBAT channel.  
0: VBAT channel disabled  
1: VBAT channel enabled  
*Note:* The software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

**Bit 23 TSEN**: Temperature sensor voltage enable  
This bit is set and cleared by software to control VSENSE channel.  
0: Temperature sensor channel disabled  
1: Temperature sensor channel enabled  
*Note:* The software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

**Bit 22 VREFEN**: VREFINT enable  
This bit is set and cleared by software to enable/disable the VREFINT channel.  
0: VREFINT channel disabled  
1: VREFINT channel enabled  
*Note:* The software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).
Bits 21:18 **PRESC[3:0]: ADC prescaler**

These bits are set and cleared by software to select the frequency of the clock to the ADC. The clock is common for all the ADCs.

- 0000: input ADC clock not divided
- 0001: input ADC clock divided by 2
- 0010: input ADC clock divided by 4
- 0011: input ADC clock divided by 6
- 0100: input ADC clock divided by 8
- 0101: input ADC clock divided by 10
- 0110: input ADC clock divided by 12
- 0111: input ADC clock divided by 16
- 1000: input ADC clock divided by 32
- 1001: input ADC clock divided by 64
- 1010: input ADC clock divided by 128
- 1011: input ADC clock divided by 256
- Others: Reserved, must not be used

*Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0). The ADC prescaler value is applied only when CKMODE[1:0] = 0b00.*

Bits 17:16 **CKMODE[1:0]: ADC clock mode**

These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):

- 00: CK_ADCx (x=1 to 23) (Asynchronous clock mode), generated at product level (refer to Section Reset and Clock Control (RCC))
- 01: adc_sclk/1 (Synchronous clock mode).
- 10: adc_sclk/2 (Synchronous clock mode)
- 11: adc_sclk/4 (Synchronous clock mode)

Whatever CKMODE[1:0] settings, an additional divider factor of 2 is applied to the clock delivered to the analog ADC block.

In synchronous clock mode, when adc_ker_ck = 2 x adc_hclk, there is no jitter in the delay from a timer trigger to the start of a conversion.

*Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).*

Bits 15:14 **DAMDF[1:0]: Dual ADC Mode Data Format**

This bit-field is set and cleared by software. It specifies the data format in the common data register ADCx_CDR.

- 00: Dual ADC mode without data packing (ADCx_CDR and ADCx_CDR2 registers not used).
- 01: Reserved.
- 10: Data formatting mode for 32 down to 10-bit resolution
- 11: Data formatting mode for 8-bit resolution

*Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).*

Bits 13:12 Reserved, must be kept at reset value.
Bits 11:8 **DELAY[3:0]**: Delay between 2 sampling phases

These bits are set and cleared by software. These bits are used in dual interleaved modes.

Refer to **Table 200** for the value of ADC resolution versus DELAY bits values.

*Note*: The software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]**: Dual ADC mode selection

These bits are written by software to select the operating mode.

All the ADCs are independent:

00000: Independent mode

The configurations 00001 to 01001 correspond to the following operating modes: Dual mode, master and slave ADCs working together:

00001: Combined regular simultaneous + injected simultaneous mode

00010: Combined regular simultaneous + alternate trigger mode

00011: Combined Interleaved mode + injected simultaneous mode

00100: Reserved.

00101: Injected simultaneous mode only

00110: Regular simultaneous mode only

00111: Interleaved mode only

01001: Alternate trigger mode only

All other combinations are reserved and must not be programmed.

*Note*: The software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

**Table 200. DELAY bits versus ADC resolution**

<table>
<thead>
<tr>
<th>DELAY bits</th>
<th>16-bit resolution</th>
<th>14-bit resolution</th>
<th>12-bit resolution</th>
<th>10-bit resolution</th>
<th>8-bit resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1.5 * Tadc_ker_ck</td>
<td>1.5 * Tadc_ker_ck</td>
<td>1.5 * Tadc_ker_ck</td>
<td>1.5 * Tadc_ker_ck</td>
<td>1.5 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0001</td>
<td>2.5 * Tadc_ker_ck</td>
<td>2.5 * Tadc_ker_ck</td>
<td>2.5 * Tadc_ker_ck</td>
<td>2.5 * Tadc_ker_ck</td>
<td>2.5 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0010</td>
<td>3.5 * Tadc_ker_ck</td>
<td>3.5 * Tadc_ker_ck</td>
<td>3.5 * Tadc_ker_ck</td>
<td>3.5 * Tadc_ker_ck</td>
<td>3.5 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0011</td>
<td>4.5 * Tadc_ker_ck</td>
<td>4.5 * Tadc_ker_ck</td>
<td>4.5 * Tadc_ker_ck</td>
<td>4.5 * Tadc_ker_ck</td>
<td>4.5 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0100</td>
<td>5.5 * Tadc_ker_ck</td>
<td>5.5 * Tadc_ker_ck</td>
<td>5.5 * Tadc_ker_ck</td>
<td>5.5 * Tadc_ker_ck</td>
<td>5.5 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0101</td>
<td>6.5 * Tadc_ker_ck</td>
<td>6.5 * Tadc_ker_ck</td>
<td>6.5 * Tadc_ker_ck</td>
<td>6.5 * Tadc_ker_ck</td>
<td>6.5 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0110</td>
<td>7.5 * Tadc_ker_ck</td>
<td>7.5 * Tadc_ker_ck</td>
<td>6.5 * Tadc_ker_ck</td>
<td>5.5 * Tadc_ker_ck</td>
<td>4.5 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0111</td>
<td>8.5 * Tadc_ker_ck</td>
<td>7.5 * Tadc_ker_ck</td>
<td>6.5 * Tadc_ker_ck</td>
<td>5.5 * Tadc_ker_ck</td>
<td>4.5 * Tadc_ker_ck</td>
</tr>
<tr>
<td>1000</td>
<td>8.5 * Tadc_ker_ck</td>
<td>7.5 * Tadc_ker_ck</td>
<td>6.5 * Tadc_ker_ck</td>
<td>5.5 * Tadc_ker_ck</td>
<td>4.5 * Tadc_ker_ck</td>
</tr>
<tr>
<td>others: reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
27.7.3  ADC x common regular data register for dual mode (ADCx_CDR) (x=1/2)

Address offset: 0x0C
Reset value: 0x0000 0000

The address offset is relative to the master ADC base address + 0x300.

ADC1 and ADC2 are controlled by the same interface.

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>RDATA_SLV[15:0]: Regular data of the slave ADC</th>
</tr>
</thead>
<tbody>
<tr>
<td>In dual mode, these bits contain the regular data of the slave ADC. Refer to Section 27.4.32: Dual ADC modes.</td>
<td></td>
</tr>
<tr>
<td>The data alignment is applied as described in Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE))</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>RDATA_MST[15:0]: Regular data of the master ADC.</th>
</tr>
</thead>
<tbody>
<tr>
<td>In dual mode, these bits contain the regular data of the master ADC. Refer to Section 27.4.32: Dual ADC modes.</td>
<td></td>
</tr>
<tr>
<td>The data alignment is applied as described in Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE))</td>
<td></td>
</tr>
<tr>
<td>In MDMA=0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].</td>
<td></td>
</tr>
</tbody>
</table>

27.7.4  ADC x common regular data register for 32-bit dual mode (ADCx_CDR2) (x=1/2)

Address offset: 0x10
Reset value: 0x0000 0000

The address offset is relative to the master ADC base address + 0x300.

ADC1 and ADC2 are controlled by the same interface.

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>RDATA_ALT[31:16]:</th>
</tr>
</thead>
<tbody>
<tr>
<td>In dual mode, these bits contain the regular data of the slave ADC. Refer to Section 27.4.32: Dual ADC modes.</td>
<td></td>
</tr>
<tr>
<td>The data alignment is applied as described in Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE))</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>RDATA_ALT[15:0]:</th>
</tr>
</thead>
<tbody>
<tr>
<td>In dual mode, these bits contain the regular data of the master ADC. Refer to Section 27.4.32: Dual ADC modes.</td>
<td></td>
</tr>
<tr>
<td>The data alignment is applied as described in Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE))</td>
<td></td>
</tr>
</tbody>
</table>
Analog-to-digital converters (ADC) RM0455

Bits 31:0 RDATA_ALT[31:0]: Regular data of the master/slave alternated ADCs

In dual mode, these bits alternatively contains the regular 32-bit data of the master and the slave ADC. Refer to Section 27.4.32: Dual ADC modes.

The data alignment is applied as described in Section: Data register; data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE).

27.8 ADC register map

The following table summarizes the ADC registers.

Table 201. ADC global register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000 - 0x0D0</td>
<td>Master ADC1</td>
</tr>
<tr>
<td>0x0D4 - 0x0FC</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x100 - 0x1D0</td>
<td>Slave ADC2</td>
</tr>
<tr>
<td>0x1D4 - 0x2FC</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x300 - 0x310</td>
<td>Master and slave ADCs common registers (ADC1/2)</td>
</tr>
</tbody>
</table>

Table 202. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>ADC_ISR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x04</td>
<td>ADC_IER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x08</td>
<td>ADC_CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0C</td>
<td>ADC_CFGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x10</td>
<td>ADC_CFGR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x14</td>
<td>ADC_SMPR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Table 202. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC) (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x18</td>
<td>ADC_SMPR2</td>
<td>Reset value 00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x1C</td>
<td>ADC_PCSEL</td>
<td>Reset value 00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x20</td>
<td>ADC_LTR1</td>
<td>Reset value 00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x24</td>
<td>ADC_HTR1</td>
<td>Reset value 11111111111111111111111111111111</td>
</tr>
<tr>
<td>0x28</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x2C</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x30</td>
<td>ADC_SQR1</td>
<td>Reset value 00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x34</td>
<td>ADC_SQR2</td>
<td>Reset value 00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x38</td>
<td>ADC_SQR3</td>
<td>Reset value 00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x3C</td>
<td>ADC_SQR4</td>
<td>Reset value 00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x40</td>
<td>ADC_DR</td>
<td>Reset value 00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x44</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x48</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x4C</td>
<td>ADC_JSQR</td>
<td>Reset value 00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x50</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x54</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x60</td>
<td>ADC_OFR1</td>
<td>Reset value 00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x64</td>
<td>ADC_OFR2</td>
<td>Reset value 00000000000000000000000000000000</td>
</tr>
</tbody>
</table>
### Table 202. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC) (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Offset Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x68</td>
<td>ADC_OFR3</td>
<td>SSATE, OFFSET_L_CH[4:0], OFFSET_L[25:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x6C</td>
<td>ADC_OFR4</td>
<td>SSATE, OFFSET_L_CH[4:0], OFFSET_L[25:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x70-0x7C</td>
<td>Reserved</td>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td>0x80</td>
<td>ADC_JDR1</td>
<td>JDATA1[31:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x84</td>
<td>ADC_JDR2</td>
<td>JDATA2[31:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x88</td>
<td>ADC_JDR3</td>
<td>JDATA3[31:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8C</td>
<td>ADC_JDR4</td>
<td>JDATA4[31:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8C-0x9C</td>
<td>Reserved</td>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td>0xA0</td>
<td>ADC_AWD2CR</td>
<td>AWD2CH[19:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xA4</td>
<td>ADC_AWD3CR</td>
<td>AWD3CH[19:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x88-0xAC</td>
<td>Reserved</td>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td>0xB0</td>
<td>ADC_LTR2</td>
<td>LTR2[25:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xB4</td>
<td>ADC_HTR2</td>
<td>HTR2[25:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xB8</td>
<td>ADC_LTR3</td>
<td>LTR3[25:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xBC</td>
<td>ADC_HTR3</td>
<td>HTR3[25:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xC0</td>
<td>ADC_DIFSEL</td>
<td>DIFSEL[19:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xC4</td>
<td>ADC_CALFACT</td>
<td>CALFACT_D[10:0], CALFACT_S[10:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xC8</td>
<td>ADC_CALFACT2</td>
<td>LINCALFACT[29:0]</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 202. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC) (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xCC - 0xD0 | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Table 203. ADC register map and reset values (master and slave ADC common registers) offset =0x300

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | ADCx_CSR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0x04 Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | ADCx_CCR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | ADCx_CDR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | ADCx_CDR2     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.3 on page 128 for the register boundary addresses.
28 Digital temperature sensor (DTS)

28.1 Introduction

The device embeds a sensor that converts the temperature into a square wave which frequency is proportional to the temperature. The frequency is measured either with the PCLK or the LSE clock.

28.2 DTS main features

The temperature sensor block main features are the following:

- Start of measurement triggered by software or 4 hardware sources
- Programmable sampling time to increase temperature measurement precision
- Counter synchronized on LSE or PCLK clock
- Temperature watchdog on low and high threshold
- Interrupt generation when the temperature is lower or higher than predefined thresholds and at the end of measurement.
- Asynchronous wakeup signal generation when the temperature is higher/lower than a predefined threshold (LSE mode only)
28.3 DTS functional description

28.3.1 DTS block diagram

The temperature sensor block diagram is shown in Figure 223.

Figure 223. Temperature sensor functional block diagram

28.3.2 DTS internal signals

Table 204. DTS internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dts_lse</td>
<td>Digital input</td>
<td>LSE clock</td>
</tr>
<tr>
<td>dts_pclk</td>
<td>Digital input</td>
<td>APB clock</td>
</tr>
<tr>
<td>dts_it</td>
<td>Digital output</td>
<td>Temperature sensor interrupt</td>
</tr>
<tr>
<td>dts_wkup</td>
<td>Digital output</td>
<td>Temperature sensor wakeup</td>
</tr>
</tbody>
</table>
28.3.3 DTS block operation

The analog part of the temperature sensor outputs a frequency that is proportional to the absolute temperature (CLK_PTAT). The frequency measurement is based on the PCLK or the LSE clock.

Before each measurement, the temperature sensor performs a calibration of the frequency generation blocks.

28.3.4 Operating modes

Several operating modes can be selected by setting the REFCLK_SEL bit in Temperature sensor configuration register 1 (DTS_CFGR1):

- PCLK only (REFCLK_SEL = 0)
  
  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using PCLK clock

- PCLK and LSE (REFCLK_SEL = 1)
  
  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using the LSE clock.

- LSE only (REFCLK_SEL = 1) and PCLK OFF
  
  The registers cannot be accessed. The measurement can be performed using the LSE clock. This mode is used to exit from Sleep mode by using hardware triggers and the asynchronous interrupt line.

28.3.5 Calibration

The temperature sensor must run the calibration prior to any frequency measurement. The calibration is performed automatically when the temperature measurement is triggered.

28.3.6 Prescaler

When a calibration is ongoing, the counter clock must be slower than 1 MHz. This is achieved by the PCLK clock prescaler embedded in the temperature sensor.

During the temperature measurement period, the prescaler is bypassed.

- When PCLK is used as reference clock (REFCLK_SEL set to 0 in DTS_CFGR1), a prescaler is used. Its division ratio must be configured up to 127 (refer to the HSREF_CLK_DIV[6:0] register definition for the divider setting).
- When LSE is used as reference clock (REFCLK_SEL set to 1 in DTS_CFGR1), the timebase is equal to 2 LSE periods. In this case, no prescaler is used.
28.3.7 Temperature measurement principles

The analog part of temperature sensor outputs a signal (CLK_PTAT) which FM(T) frequency is temperature-dependent (typically 641 kHz).

Either PCLK or LSE can be selected as reference clock (REF_CLK) through the REFCLK_SEL bit in DTS_CFGR1.

The counting method depends on the REF_CLK frequency. This is due to the fact that two counters are implemented in the temperature sensor block:

- For low REF_CLK frequencies, a counting of FM(T) cycles is performed during one or several REF_CLK cycles.
- For high REF_CLK frequencies, a counting of REF_CLK cycles is performed during one or several FM(T) cycles.

This counter behavior is shown in Figure 224 and Figure 225.

**Figure 224. Method for low REF_CLK frequencies**

![Figure 224](MSv40361V1)

1. To increase the precision, FM(T) measurement can be done on several LSE periods.

**Figure 225. Method for high REF_CLK frequencies**

![Figure 225](MSv40850V1)

1. To increase the precision, PCLK measurement can be done on several FM(T) periods.

The counting result is stored in the DTS_DR register (see Temperature sensor data register (DTS_DR)).

Once the FM(T) frequency has been obtained, the corresponding temperature can be calculated by software using the following formula:

- When PCLK is used:

  \[ T = T_0 + \left( \frac{F_{PCLK}}{TS1_MFREQ} \times TS1_SMP\_TIME - 100 \times TS1\_FMT0} / TS1\_RAMP\_COEFF \right) \]

  where

  - T0 (factory calibration temperature) is equal to 30 °C.
  - TS1_FMT0 is measured and stored in the DTS_T0VALR1 register. It is expressed in hundreds of Hertz.
TS1_RAMP_COEFF is measured during tests in factory and stored in DTS_RAMPVALR register. This value is expressed in Hz/°C.

- When the LSE clock is used

\[ T = T_0 + \left( \frac{F_{LSE} \times TS1_MFREQ / TS1_SMP_TIME - (100 \times TS1_FMT0)}{TS1_RAMP_COEFF} \right) \]

### 28.3.8 Sampling time

The sampling period can be increased to improve measurement accuracy. This is useful when the reference frequency (REF_CLK) is close to the FM(T) frequency. The default value is one REF_CLK cycle in LSE mode, and one FM(T) cycle in PCLK mode.

The sampling time is configured through TS1_SMP_TIME bits in DTS_CFRG1 register (see Table 205).

<table>
<thead>
<tr>
<th>TS1_SMP_TIME[3:0]</th>
<th>LSE or FM(T) clock cycle(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
</tr>
</tbody>
</table>
28.3.9 Trigger input

The temperature measurement can be triggered either by software or by an external event. The trigger source can be selected through TS1_INTRIG[3:0] bits in DTS_CFRG1.

Table 206. Trigger configuration

<table>
<thead>
<tr>
<th>Name</th>
<th>TS1_INTRIG[3:0]</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N.A</td>
<td>0 0 0 0</td>
<td>No hardware trigger</td>
</tr>
<tr>
<td>ts1_trg0</td>
<td>0 0 0 1</td>
<td>LPTIMER1</td>
</tr>
<tr>
<td>ts1_trg1</td>
<td>0 0 1 0</td>
<td>LPTIMER2</td>
</tr>
<tr>
<td>ts1_trg2</td>
<td>0 1 0 0</td>
<td>EXTI13</td>
</tr>
<tr>
<td>ts1_trg3</td>
<td>0 1 0 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg4</td>
<td>0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg5</td>
<td>0 1 1 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg6</td>
<td>0 1 1 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg7</td>
<td>1 0 0 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg8</td>
<td>1 0 0 1</td>
<td>Reserved</td>
</tr>
<tr>
<td>ts1_trg9</td>
<td>1 0 1 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg10</td>
<td>1 0 1 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg11</td>
<td>1 1 0 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg12</td>
<td>1 1 0 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg13</td>
<td>1 1 1 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg14</td>
<td>1 1 1 1</td>
<td></td>
</tr>
</tbody>
</table>

Note: Hardware triggers are active only on the rising edge. The temperature sensor can only capture a hardware trigger rising edge when TS1_RDY bit is set (see Section 28.3.10: On-off control and ready flag, otherwise the trigger is ignored. If a trigger source changes on-the-fly, the new trigger source signal should be low. If the new source signal is high, the temperature sensor detects a rising edge and start the measurement sequence.

28.3.10 On-off control and ready flag

The DTS block can be enabled by setting TS1_EN bit in DTS_CFRG1 register. The TS1_RDY flag in the Temperature sensor status register (DTS_SR) indicate that the DTS block is ready for temperature measurement: when TS1_RDY bit is set to 1, the measurement can be started. Once a measurement has started, TS1_RDY bit is reset. New measurement requests will then be ignored. Once the measurement is finished, TS1_RDY bit is set again to indicate the sensor is ready to start a new measurement.
28.3.11 Temperature measurement sequence

Start of measurement can be triggered by software or hardware.

**Software trigger**

The software trigger is selected when TS1_INTRIG_SEL[3:0] is set to '0000' in DTS_CFRGR1.

- If TS1_RDY is set to 1, writing TS1_START bit to 1 in DTS_CFRGR1 starts the measurement.
- If TS1_RDY equals 0, the software trigger does not start until TS1_RDY is set.
- If TS1_START bit is kept at 1 once the measurement is finished, then the TS1_RDY flag becomes 1 and the measurement restarts.

**Hardware trigger**

TS1_INTRIG_SEL[3:0] bits allow selecting one hardware trigger out of 4. If TS1_RDY is set to 1, a rising edge on the trigger signal starts the measurement. When TS1_RDY is 0, the rising edge is ignored.

**Measurement sequence**

One measurement contains two steps: the calibration of the analog blocks and the measurement. The calibration automatically starts when the measurement is triggered (see Section 28.3.5: Calibration). The measurement period depends on the following DTS_CFRGR1 bits:

- the reference clock selected through REFCLK_SEL bit
- the divider ratio configured by HSREF_CLK_DIV bits
- the sampling time defined by TS1_SMP_TIME bits.

![Figure 226. Temperature sensor sequence](MSv40363V1)
28.4 DTS low-power modes

Table 207. Temperature sensor behavior in low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>Only works in LSE mode. DTS interrupt causes the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>Only works in LSE mode. DTS interrupt cause the device to exit from Stop mode.</td>
</tr>
</tbody>
</table>

28.5 DTS interrupts

There are two ways to use the DTS block as an interrupt source. The DTS interrupt line can be connected to the CPU NVIC (see Section 28.5.2: Synchronous interrupt) or to the EXTI controller (see Section 28.5.3: Asynchronous wakeup).

28.5.1 Temperature window comparator

The DTS_ITR1 register allows defining the high and low threshold that will be used for temperature comparison. If the temperature data is equal or higher than TS1_HITTHD, or equal or lower than TS1_LITTHD bit, an interrupt is generated and the corresponding flag, TS1_ITLF, TS1_ITHF, TS1_AITLF and TS1_AITHF, is set in the DTS_SR register (see Section 28.6.6).

28.5.2 Synchronous interrupt

A global interrupt output line is available on the DTS block. The interrupt can be generated at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see Section 28.5.1: Temperature window comparator).

Three interrupt events can be select via 3 bits in DTS_ITENR register (see Section 28.6.7). All combinations of interrupts are allowed.

The TS1_ITEF, TS1_ITLF and TS1_ITHF flags in the DTS_SR register reflect the interrupt event. They can be reset with the correspond bits of the DTS_ICIFR register (see Section 28.6.8).

28.5.3 Asynchronous wakeup

The DTS block also provides an asynchronous interrupt line. It is used only when the LSE is selected as reference clock (REFCLK_SEL=1).

This line can generate a signal that wakes up the system from Sleep mode at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see Section 28.5.1: Temperature window comparator).

Three asynchronous wakeup events can be selected via 3 bits in DTS_ITENR register. All combination of interrupts are allowed.
The TS1_AITEF, TS1_AITLF and TS1_AITHF flags in the DTS_SR register reflect the interrupt status. They can be reset with the correspond bits of the DTS_ICIFR register.

The following table shows the interrupt bits and their description.

### Table 208. Interrupt control bits

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Interrupt flag</th>
<th>Enable control bit</th>
<th>Interrupt clear bit</th>
<th>Exit from Sleep mode</th>
<th>Synchronous/Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the end of measurement</td>
<td>TS1_ITEF in DTS_SR</td>
<td>TS1_ITEEN in DTS_ITENR</td>
<td>TS1_CITEF in DTS_ICIFR</td>
<td>NO</td>
<td>Synchronous on PCLK</td>
</tr>
<tr>
<td>When the measure is equal or exceeds the low threshold</td>
<td>TS1_ITLF in DTS_SR</td>
<td>TS1_ITLEN in DTS_ITENR</td>
<td>TS1_CITLF in DTS_ICIFR</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>When the measure is equal or exceeds the high threshold</td>
<td>TS1_ITHF in DTS_SR</td>
<td>TS1_ITHEN in DTS_ITENR</td>
<td>TS1_CITHF in DTS_ICIFR</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>At the end of measurement</td>
<td>TS1_AITEF in DTS_SR</td>
<td>TS1_AITEEN in DTS_ITENR</td>
<td>TS1_CAITEF in DTS_ICIFR</td>
<td>YES</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>When the measure is equal or exceeds the low threshold</td>
<td>TS1_AITLF in DTS_SR</td>
<td>TS1_AITLEN in DTS_ITENR</td>
<td>TS1_CAITLF in DTS_ICIFR</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>When the measure is equal or exceeds the high threshold</td>
<td>TS1_AITHF in DTS_SR</td>
<td>TS1_AITHEN in DTS_ITENR</td>
<td>TS1_CAITHF in DTS_ICIFR</td>
<td>YES</td>
<td></td>
</tr>
</tbody>
</table>
28.6 **DTS registers**

The registers of this peripheral can only be accessed by-word (32-bit).

### 28.6.1 Temperature sensor configuration register 1 (DTS_CFRG1)

DTS_CFRG1 is the configuration register for temperature sensor 1.

**Address offset:** 0x00

**System reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>30:24</td>
<td>HSRF_CLK_DIV[6:0]</td>
<td>High speed clock division ratio</td>
</tr>
<tr>
<td>23</td>
<td>REFCLK_SEL</td>
<td>Reference clock selection bit</td>
</tr>
<tr>
<td>21:16</td>
<td>TS1_SMP_TIME[3:0]</td>
<td>Sampling time for temperature sensor 1</td>
</tr>
<tr>
<td>15:8</td>
<td>TS1_INTRIG_SEL[3:0]</td>
<td>Input trigger selection bit for temperature sensor 1</td>
</tr>
<tr>
<td>7:0</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
</tbody>
</table>

**Bits 30:24 HSRF_CLK_DIV[6:0]:** High speed clock division ratio

These bits are set and cleared by software. They can be used to define the division ratio for the main clock in order to obtain the internal frequency lower than 1 MHz required for the calibration. They are applicable only for calibration when PCLK is selected as reference clock (REFCLK_SEL=0).

- 0000000: No divider
- 0000001: 1/1 divider
- 0000010: 1/2 division ratio
- ...
- 1111111: 1/127 division ratio

**Bits 23:21 Reserved:** Must be kept at reset value.

**Bit 20 REFCLK_SEL:** Reference clock selection bit

This bit is set and cleared by software. It indicates whether the reference clock is the high speed clock (PCLK) or the low speed clock (LSE).

- 0: High speed reference clock (PCLK)
- 1: Low speed reference clock (LSE)

**Bits 19:16 TS1_SMP_TIME[3:0]:** Sampling time for temperature sensor 1

These bits allow increasing the sampling time to improve measurement precision.

When the PCLK clock is selected as reference clock (REFCLK_SEL = 0), the measurement will be performed at TS1_SMP_TIME period of CLK_PTAT.

When the LSE is selected as reference clock (REFCLK_SEL =1), the measurement will be performed at TS1_SMP_TIME period of LSE.

**Bits 15:12 Reserved:** Must be kept at reset value.

**Bits 11:8 TS1_INTRIG_SEL[3:0]:** Input trigger selection bit for temperature sensor 1

These bits are set and cleared by software. They select which input triggers a temperature measurement. Refer to *Table 206: Trigger configuration*.

**Bits 7:5 Reserved:** Must be kept at reset value.
Bit 4 **TS1_START**: Start frequency measurement on temperature sensor 1
This bit is set and cleared by software.
0: No software trigger.
1: Software trigger for a frequency measurement. (only if TS1 is ready).

Bits 3:1 **Reserved**, must be kept at reset value.

Bit 0 **TS1_EN**: Temperature sensor 1 enable bit
This bit is set and cleared by software.
0: Temperature sensor 1 disabled
1: Temperature sensor 1 enabled

**Note**: Once enabled, the temperature sensor is active after a specific delay time. The TS1_RDY flag will be set when the sensor is ready.

### 28.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1)

DTS_T0VALR1 contains the value of the factory calibration temperature (T0) for temperature sensor 1. The system reset value is factory trimmed.

Address offset: 0x08

System reset value: 0x000X XXXX

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

**TS1_FMT0[15:0]**

Bits 31:18 **Reserved**, must be kept at reset value.

Bits 17:16 **TS1_T0[1:0]**: Engineering value of the T0 temperature for temperature sensor 1.
00: 30 °C
01: 110 °C
Others: Reserved, must not be used.

Bits 15:0 **TS1_FMT0[15:0]**: Engineering value of the frequency measured at T0 for temperature sensor 1
This value is expressed in 0.1 kHz.

### 28.6.3 Temperature sensor ramp value register (DTS_RAMPVALR)
The DTS_RAMPVALR is the ramp coefficient for the temperature sensor. The system reset value is factory trimmed.

Address offset: 0x10

System reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>
28.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1)

DTS_ITR1 contains the threshold values for sensor 1.

Address offset: 0x14

System reset value: 0x0000 0000

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

Bits 31:16 **TS1_HITTHD[15:0]**: High interrupt threshold for temperature sensor 1

These bits are set and cleared by software. They indicate the highest value than can be reached before raising an interrupt signal.

Bits 15:0 **TS1_LITTHD[15:0]**: High interrupt threshold for temperature sensor 1

These bits are set and cleared by software. They indicate the lowest value than can be reached before raising an interrupt signal.

28.6.5 Temperature sensor data register (DTS_DR)

The DTS_DR contains the number of REF_CLK cycles used to compute the FM(T) frequency.

Address offset: 0x1C

System reset value: 0x0000 0000

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

Bits 31:16 **TS1_MFREQ[15:0]**: Value of the counter output value for temperature sensor 1

Bits 15:0 **TS1_MFREQ[15:0]**: Value of the counter output value for temperature sensor 1

Reserved, must be kept at reset value.
28.6.6 Temperature sensor status register (DTS_SR)

Address offset: 0x20

System reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

**Bit 15 TS1_RDY**: Temperature sensor 1 ready flag

This bit is set and reset by hardware.
It indicates that a measurement is ongoing.
0: Temperature sensor 1 busy
1: Temperature sensor 1 ready

Bits 14:7 Reserved, must be kept at reset value.

**Bit 6 TS1_AITHF**: Asynchronous interrupt flag for high threshold on temperature sensor 1

This bit is set by hardware when the high threshold is reached.
It is cleared by software by writing 1 to the TS1_CAITHF bit in the DTS_ICIFR register.
0: High threshold not reached on temperature sensor 1
1: High threshold reached on temperature sensor 1

*Note: This bit is active only when the TS1_AITHFEN bit is set*

**Bit 5 TS1_AITLF**: Asynchronous interrupt flag for low threshold on temperature sensor 1

This bit is set by hardware when the low threshold is reached.
It is cleared by software by writing 1 to the TS1_CAITLF bit in the DTS_ICIFR register.
0: Low threshold not reached on temperature sensor 1
1: Low threshold reached on temperature sensor 1

*Note: This bit is active only when the TS1_AITLFEN bit is set*

**Bit 4 TS1_AITEF**: Asynchronous interrupt flag for end of measure on temperature sensor 1

This bit is set by hardware when a temperature measure is done.
It is cleared by software by writing 1 to the TS1_CAITEF bit in the DTS_ICIFR register.
0: End of measure not detected on temperature sensor 1
1: End of measure detected on temperature sensor 1

*Note: This bit is active only when the TS1_AITEFEN bit is set*

**Bit 3** Reserved, must be kept at reset value.
Bit 2 **TS1ITHF**: Interrupt flag for high threshold on temperature sensor 1, synchronized on PCLK.  
This bit is set by hardware when the high threshold is set and reached.  
It is cleared by software by writing 1 to the TS1_CITHF bit in the DTS_ICIFR register.  
0: High threshold not reached on temperature sensor 1  
1: High threshold reached on temperature sensor 1  
*Note: This bit is active only when the TS1ITHFEN bit is set*

Bit 1 **TS1ITLF**: Interrupt flag for low threshold on temperature sensor 1, synchronized on PCLK.  
This bit is set by hardware when the low threshold is set and reached.  
It is cleared by software by writing 1 to the TS1_CITLF bit in the DTS_ICIFR register.  
0: Low threshold not reached on temperature sensor 1  
1: Low threshold reached on temperature sensor 1  
*Note: This bit is active only when the TS1ITLFEN bit is set*

Bit 0 **TS1ITEF**: Interrupt flag for end of measurement on temperature sensor 1, synchronized on PCLK.  
This bit is set by hardware when a temperature measure is done.  
It is cleared by software by writing 1 to the TS2_CITEF bit in the DTS_ICIFR register.  
0: No end of measurement detected on temperature sensor 1  
1: End of measure detected on temperature sensor 1  
*Note: This bit is active only when the TS1ITEFEN bit is set*

### 28.6.7 Temperature sensor interrupt enable register (DTS_ITENR)

Address offset: 0x24  
System reset value: 0x0000 0000

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1AITHEN**: Asynchronous interrupt enable flag on high threshold for temperature sensor 1.  
This bit are set and cleared by software.  
It enables the asynchronous interrupt when the temperature is above the high threshold (only when REFCLK_SEL = 1’b1)  
0: Asynchronous interrupt on high threshold disabled for temperature sensor 1  
1: Asynchronous interrupt on high threshold enabled for temperature sensor 1

Bit 5 **TS1AITLEN**: Asynchronous interrupt enable flag for low threshold on temperature sensor 1.  
This bit are set and cleared by software.  
It enables the asynchronous interrupt when the temperature is below the low threshold (only when REFCLK_SEL = 1)  
0: Asynchronous interrupt on low threshold disabled for temperature sensor 1  
1: Asynchronous interrupt on low threshold enabled for temperature sensor 1
Bit 4 **TS1_AITEEN**: Asynchronous interrupt enable flag for end of measurement on temperature sensor 1

This bit is set and cleared by software.
It enables the asynchronous interrupt for end of measurement (only when REFCLK_SEL = 1).
0: Asynchronous interrupt for end of measurement disabled on temperature sensor 1
1: Asynchronous interrupt for end of measurement enabled on temperature sensor 1

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_ITHEN**: Interrupt enable flag for high threshold on temperature sensor 1, synchronized on PCLK.

This bit is set and cleared by software.
It enables the interrupt when the measure reaches or is above the high threshold.
0: Synchronous interrupt for high threshold disabled on temperature sensor 1
1: Synchronous interrupt for high threshold enabled on temperature sensor 1

Bit 1 **TS1_ITLEN**: Interrupt enable flag for low threshold on temperature sensor 1, synchronized on PCLK.

This bit is set and cleared by software.
It enables the synchronous interrupt when the measure reaches or is below the low threshold.
0: Synchronous interrupt for low threshold disabled on temperature sensor 1
1: Synchronous interrupt for low threshold enabled on temperature sensor 1

Bit 0 **TS1_ITEEN**: Interrupt enable flag for end of measurement on temperature sensor 1, synchronized on PCLK.

This bit is set and cleared by software.
It enables the synchronous interrupt for end of measurement.
0: Synchronous interrupt for end of measurement disabled on temperature sensor 1
1: Synchronous interrupt for end of measurement enabled on temperature sensor 1

**28.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR)**

DTS_ICIFR is the control register for the interrupt flags.

Address offset: 0x28

System reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1_CAITHF**: Asynchronous interrupt clear flag for high threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITHF flag in the DTS_SR register.

Bit 5 **TS1_CAITLF**: Asynchronous interrupt clear flag for low threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITLF flag in the DTS_SR register.
Bit 4 **TS1_CAITEF**: Write once bit. Clear the asynchronous IT flag for End Of Measure for thermal sensor 1.

Writing 1 clears the TS1_AITEF flag of the DTS_SR register.

Bit 3 **Reserved**, must be kept at reset value.

Bit 2 **TS1_CITHF**: Interrupt clear flag for high threshold on temperature sensor 1

Writing this bit to 1 clears the TS1ITHF flag in the DTS_SR register.

Bit 1 **TS1_CITLF**: Interrupt clear flag for low threshold on temperature sensor 1

Writing 1 to this bit clears the TS1_ITLF flag in the DTS_SR register.

Bit 0 **TS1_CITEF**: Interrupt clear flag for end of measurement on temperature sensor 1

Writing 1 to this bit clears the TS1_ITEF flag in the DTS_SR register.

### 28.6.9 Temperature sensor option register (DTS_OR)

The DTS_OR contains general-purpose option bits.

Address offset: 0x2C

System reset value: 0x0000 0000

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

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

Bits 31:0 **TS_OP[31:0]**: general purpose option bits
28.6.10 DTS register map

The following table summarizes the temperature sensor registers.

| Offset | Register   | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | DTS_CFGR1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        |            | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x08   | DTS_T0VALR1 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        |            | Reset value | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  |
| 0x10   | DTS_RAMPVALR | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        |            | Reset value | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  |
| 0x14   | DTS_ITR    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        |            | Reset value | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  | X  |
| 0x18   | DTS_DR     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        |            | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x20   | DTS_SR     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        |            | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x24   | DTS_ITENR  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        |            | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x28   | DTS_ICIFR  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        |            | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x2C   | DTS_OR     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        |            | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 on page 128 for the register boundary addresses.
29  Digital-to-analog converter (DAC)

29.1  Introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. The DAC features up to two output channels, each with its own converter. In dual DAC channel mode, conversions could be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, VREF+ (shared with others analog peripherals) is available for better resolution. An internal reference can also be set on the same input. Refer to voltage reference buffer (VREFBUF) section.

The DAC_OUTx pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on chip peripheral. The DAC output buffer can be optionally enabled to allow a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the Sample and hold mode.

29.2  DAC main features

The DAC main features are the following (see Figure 227: Dual-channel DAC block diagram)

- Two DAC interfaces, maximum two output channels each
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Each DAC output can be disconnected from the DAC_OUTx output pin
- DAC output connection to on chip peripherals
- Sample and hold mode for low power operation in Stop mode
- Input voltage reference from VREF+ pin or internal VREFBUF reference

Figure 227 shows the block diagram of a DAC channel and Table 211 gives the pin description.
29.3 DAC implementation

Table 210. DAC implementation

<table>
<thead>
<tr>
<th>DAC features</th>
<th>DAC1</th>
<th>DAC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual channel</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Output buffer</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>I/O connection</td>
<td>DAC1_OUT1 on PA4, DAC1_OUT2 on PA5</td>
<td>DAC2_OUT1 on PA6</td>
</tr>
<tr>
<td>Maximum sampling time</td>
<td></td>
<td>1MSPS</td>
</tr>
</tbody>
</table>
29.4 DAC functional description

29.4.1 DAC block diagram

Figure 227. Dual-channel DAC block diagram

1. MODEx bits in the DAC_MCR control the output mode and allow switching between the Normal mode in buffer/unbuffered configuration and the Sample and hold mode.

2. Refer to Section 29.3: DAC implementation for channel2 availability.

3. DAC channel2 is available only on DAC1.
29.4.2 DAC pins and internal signals

The DAC includes:

- Up to two output channels
- The DAC_OUTPUTx can be disconnected from the output pin and used as an ordinary GPIO
- The dac_outx can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered
- Sample and hold block and registers operational in Stop mode, using the LSI clock source (dac_hold_ck) for static conversion.

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DAC_OUTPUTx output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The Sample and hold block and its associated registers can run in Stop mode using the LSI clock source (dac_hold_ck).

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREF+</td>
<td>Input, analog reference positive</td>
<td>The higher/positive reference voltage for the DAC, $V_{REF^+} \leq V_{DDA_{max}}$ (refer to datasheet)</td>
</tr>
<tr>
<td>VDDA</td>
<td>Input, analog supply</td>
<td>Analog power supply</td>
</tr>
<tr>
<td>VSSA</td>
<td>Input, analog supply ground</td>
<td>Ground for analog power supply</td>
</tr>
<tr>
<td>DAC_OUTPUTx</td>
<td>Analog output signal</td>
<td>DAC channelx analog output</td>
</tr>
</tbody>
</table>

### Table 212. DAC internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dac_ch1_dma</td>
<td>Bidirectional</td>
<td>DAC channel1 DMA request/acknowledge</td>
</tr>
<tr>
<td>dac_ch2_dma</td>
<td>Bidirectional</td>
<td>DAC channel2 DMA request/acknowledge</td>
</tr>
<tr>
<td>dac_ch1_trgx (x = 1 to 15)</td>
<td>Inputs</td>
<td>DAC channel1 trigger inputs</td>
</tr>
<tr>
<td>dac_ch2_trgx (x = 1 to 15)</td>
<td>Inputs</td>
<td>DAC channel2 trigger inputs</td>
</tr>
<tr>
<td>dac_unr_it</td>
<td>Output</td>
<td>DAC underrun interrupt</td>
</tr>
<tr>
<td>dac_pclk</td>
<td>Input</td>
<td>DAC peripheral clock</td>
</tr>
<tr>
<td>dac_hold_ck</td>
<td>Input</td>
<td>DAC low-power clock used in Sample and hold mode</td>
</tr>
<tr>
<td>dac_out1</td>
<td>Analog output</td>
<td>DAC channel1 output for on-chip peripherals</td>
</tr>
<tr>
<td>dac_out2</td>
<td>Analog output</td>
<td>DAC channel2 output for on-chip peripherals</td>
</tr>
</tbody>
</table>
Table 213. DAC1 interconnection

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source</th>
<th>Source type</th>
</tr>
</thead>
<tbody>
<tr>
<td>dac_hold_ck</td>
<td>lsi_ck (selected in the RCC)</td>
<td>LSI clock selected in the RCC</td>
</tr>
<tr>
<td>dac_chx_trg1 (x = 1, 2)</td>
<td>tim1_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg2 (x = 1, 2)</td>
<td>tim2_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg3 (x = 1, 2)</td>
<td>tim4_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg4 (x = 1, 2)</td>
<td>tim5_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg5 (x = 1, 2)</td>
<td>tim6_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg6 (x = 1, 2)</td>
<td>tim7_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg7 (x = 1, 2)</td>
<td>tim8_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg8 (x = 1, 2)</td>
<td>tim15_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg11 (x = 1, 2)</td>
<td>lptim1_out</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg12 (x = 1, 2)</td>
<td>lptim2_out</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg13 (x = 1, 2)</td>
<td>exti9</td>
<td>External pin</td>
</tr>
<tr>
<td>dac_chx_trg14 (x = 1, 2)</td>
<td>lptim3_out</td>
<td>Internal signal from on-chip timers</td>
</tr>
</tbody>
</table>

Table 214. DAC2 interconnection

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source</th>
<th>Source type</th>
</tr>
</thead>
<tbody>
<tr>
<td>dac_hold_ck</td>
<td>lsi_ck (selected in the RCC)</td>
<td>LSI clock selected in the RCC</td>
</tr>
<tr>
<td>dac_ch1_trg1</td>
<td>tim1_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_ch1_trg2</td>
<td>tim2_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_ch1_trg3</td>
<td>tim4_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_ch1_trg4</td>
<td>tim5_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_ch1_trg5</td>
<td>tim6_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_ch1_trg6</td>
<td>tim7_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_ch1_trg7</td>
<td>tim8_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_ch1_trg8</td>
<td>tim15_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_ch1_trg11</td>
<td>lptim1_out</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_ch1_trg12</td>
<td>lptim2_out</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_ch1_trg13</td>
<td>exti9</td>
<td>External pin</td>
</tr>
<tr>
<td>dac_ch1_trg14</td>
<td>lptim3_out</td>
<td>Internal signal from on-chip timers</td>
</tr>
</tbody>
</table>
29.4.3 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a \( t_{\text{WAKEUP}} \) startup time.

*Note:* The ENx bit enables the analog DAC channel x only. The DAC channel x digital interface is enabled even if the ENx bit is reset.

29.4.4 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel
  
  There are three possibilities:
  
  - 8-bit right alignment: the software has to load data into the DAC_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)
  
  - 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)
  
  - 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

  Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DHRx (data holding registerx, which are internal non-memory-mapped registers). The DHRx register is then loaded into the DORx register either automatically, by software trigger or by an external event trigger.

- Dual DAC channels (when available)
  
  There are three possibilities:
  
  - 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)
  
  - 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)
  
  - 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

  Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC_DOR1.
and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

**Figure 229. Data registers in dual DAC channel mode**

<table>
<thead>
<tr>
<th>31</th>
<th>24</th>
<th>15</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-bit right aligned</td>
<td>12-bit left aligned</td>
<td>12-bit right aligned</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 29.4.5 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write operation to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one dac_pclk clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_pclk clock cycles after the trigger signal.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

**Figure 230. Timing diagram for conversion with trigger disabled TEN = 0**

### 29.4.6 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and $V_{REF+}$.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DACoutput} = V_{\text{REF}} \times \frac{\text{DOR}}{4096}$$
29.4.7 DAC trigger selection

If the TENx control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[3:0] control bits determine which out of 16 possible events triggers the conversion as shown in TSELx[3:0] bits of the DAC_CR register. These events can be either the software trigger or hardware triggers. Refer to the interconnection table in Section 29.4.2: DAC pins and internal signals.

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three dac_pclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: TSELx[3:0] bit cannot be changed when the ENx bit is set.
When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one dac_pclk clock cycle.

29.4.8 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC_DHRx register is transferred into the DAC_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit must be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC_DHRx to DAC_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHRx before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

The software must clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion could be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.
29.4.9 DAC noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac_pclk clock cycles after each trigger event, following a specific calculation algorithm.

Figure 231. DAC LFSR register calculation algorithm

The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

If LFSR is 0x0000, a ‘1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 232. DAC conversion (SW trigger enabled) with LFSR wave generation

Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.
29.4.10 DAC triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10”. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_pclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is transferred into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 233. DAC triangle wave generation

Figure 234. DAC conversion (SW trigger enabled) with triangle wave generation

Note: The DAC trigger must be enabled for triangle wave generation by setting the TENx bit in the DAC_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.
29.4.11 DAC channel modes

Each DAC channel can be configured in Normal mode or Sample and hold mode. The output buffer can be enabled to allow a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In Normal mode, there are four combinations, by changing the buffer state and by changing the DAC_OUTx pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:
- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:
- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

Sample and hold mode

In Sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI low-speed clock (dac_hold_ck) in addition to the dac_pclk clock, allowing to use the DAC channels in deep low power modes such as Stop mode.

The LSI low-speed clock (dac_hold_ck) must not be stopped on-the-fly when the Sample and hold mode is enabled.

The sample/hold mode operations can be divided into 3 phases:
1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEEx[9:0] bits in DAC_SHSRx register. During the write of the TSAMPLEEx[9:0] bits, the BWSTx bit in DAC_SR register is set to 1 to synchronize between both clocks domains (APB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC_SHR register
The timings for the three phases above are in units of LSI clock periods. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI ~32 KHz is selected:

- 12 cycles are required for sample phase: \( TSAMPLEx[9:0] = 11 \),
- 62 cycles are required for hold phase: \( THOLDx[9:0] = 62 \),
- and 4 cycles are required for refresh period: \( TREFRESHx[7:0] = 4 \).

In this example, the power consumption is reduced by almost a factor of 15 versus Normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

### Table 215. Sample and refresh timings

<table>
<thead>
<tr>
<th>Buffer State</th>
<th>( t_{SAMP}^{(1)(2)} )</th>
<th>( t_{REFRESH}^{(2)(3)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>( 7 \mu s + (10 \times R_{BON} \times C_{SH}) )</td>
<td>( 7 \mu s + (R_{BON} \times C_{SH}) \times \ln(2 \times N_{LSB}) )</td>
</tr>
<tr>
<td>Disable</td>
<td>( 3 \mu s + (10 \times R_{BOFF} \times C_{SH}) )</td>
<td>( 3 \mu s + (R_{BOFF} \times C_{SH}) \times \ln(2 \times N_{LSB}) )</td>
</tr>
</tbody>
</table>

1. In the above formula the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8 bits resolution, the settling time is 7 constant time.
2. \( C_{SH} \) is the capacitor in Sample and hold mode.
3. The tolerated voltage drop during the hold phase "\( V_d \)" is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires \( \ln(2 \times N_{LSB}) \) constant time of the DAC.

### Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only. Please refer to the product datasheet for product data.

\( C_{SH} = 100 \ \text{nF} \)
\( = 3.0 \ \text{V} \)

Sampling phase:
\[
t_{SAMP} = 7 \mu s + (10 \times 2000 \times 100 \times 10^{-9}) = 2.007 \ \text{ms}
\]
(where \( R_{BON} = 2 \ \text{k}\Omega \))

Refresh phase:
\[
t_{REFRESH} = 7 \mu s + (2000 \times 100 \times 10^{-9}) \times \ln(2 \times 10) = 606.1 \ \mu s
\]
(where \( N_{LSB} = 10 \) (10 LSB drop during the hold phase))

Hold phase:
\[
D_v = \frac{i_{\text{leak}} \times t_{\text{hold}}}{C_{SH}} = 0.0073 \ \text{V} \ (10 \ \text{LSB of 12bit at 3 V})
\]
\( i_{\text{leak}} = 150 \ \text{nA} \) (worst case on the IO leakage on all the temperature range)
\( t_{\text{hold}} = \frac{0.0073 \times 100 \times 10^{-9}}{(150 \times 10^{-9})} = 4.867 \ \text{ms} \)
Like in Normal mode, the Sample and hold mode has different configurations.

To enable the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:
- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:
- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals

When MODEx[2:0] bits are equal to 111, an internal capacitor, $C_{\text{int}}$, holds the voltage output of the DAC core and then drive it to on-chip peripherals.

All Sample and hold phases are interruptible, and any change in DAC_DHRx immediately triggers a new sample phase.

### Table 216. Channel output modes summary

<table>
<thead>
<tr>
<th>MODEx[2:0]</th>
<th>Mode</th>
<th>Buffer</th>
<th>Output connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>Normal mode</td>
<td>Enabled</td>
<td>Connected to external pin</td>
</tr>
<tr>
<td>0 0 1</td>
<td></td>
<td></td>
<td>Connected to external pin and to on chip-peripherals (such as comparators)</td>
</tr>
<tr>
<td>0 1 0</td>
<td></td>
<td>Disabled</td>
<td>Connected to external pin</td>
</tr>
<tr>
<td>0 1 1</td>
<td></td>
<td></td>
<td>Connected to on chip peripherals (such as comparators)</td>
</tr>
</tbody>
</table>
29.4.12 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

\[ V_{\text{out}} = \left( \frac{D}{2^N} \right) \times G \times V_{\text{ref}} + V_{\text{os}} \]

Where \( V_{\text{OUT}} \) is the analog output, \( D \) is the digital input, \( G \) is the gain, \( V_{\text{ref}} \) is the nominal full-scale voltage, and \( V_{\text{os}} \) is the offset voltage. For an ideal DAC channel, \( G = 1 \) and \( V_{\text{os}} = 0 \).

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the \( V_{\text{os}} \), a calibration is required by a trimming technique.

The calibration is only valid when the DAC channel is operating with buffer enabled (\( \text{MODE}_{x[2:0]} = 000b \) or 001b or 100b or 101b). If applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to \( V_{\text{REF}+}/2 \) signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAGx bit).

Two calibration techniques are provided:

- Factory trimming (default setting)

  The DAC buffer offset is factory trimmed. The default value of \( \text{OTRIM}_{x[4:0]} \) bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.

- User trimming

  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when \( V_{\text{DDA}} \) voltage, temperature, \( V_{\text{REF}+} \) values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the Nominal factory trimming conditions

In addition, when \( V_{\text{DD}} \) is removed (example the device enters in STANDBY or VBAT modes) the calibration is required.

The steps to perform a user trimming calibration are as below:
1. If the DAC channel is active, write 0 to ENx bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODEx[2:0] = 000b or 001b or 100b or 101b.
3. Start the DAC channelx calibration, by setting the CENx bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
   a) Write a code into OTRIMx[4:0] bits, starting by 00000b.
   b) Wait for tTRIM delay.
   c) Check if CAL_FLAGx bit in DAC_SR is set to 1.
   d) If CAL_FLAGx is set to 1, the OTRIMx[4:0] trimming code is found and can be used during device operation to compensate the output value, else increment OTRIMx[4:0] and repeat sub-steps from (a) to (d) again.

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way.

The commutation/toggle of CAL_FLAGx bit indicates that the offset is correctly compensated and the corresponding trim code must be kept in the OTRIMx[4:0] bits in DAC_CCR register.

Note: A tTRIM delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL_FLAGx bit in DAC_SR register in order to get a correct value. This parameter is specified into datasheet electrical characteristics section.

If VDDA, VREF+ and temperature conditions do not change during device operation while it enters more often in standby and VBAT mode, the software may store the OTRIMx[4:0] bits found in the first user calibration in the flash or in back-up registers, then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time. When CENx bit is set, it is not allowed to set ENx bit.

29.4.13 Dual DAC channel conversion modes (if dual channels are available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

11 conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

All modes are described in the paragraphs below.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:
1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).
When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_pclk clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three dac_pclk clock cycles later).

**Independent trigger with single LFSR generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). Then the LFSR2 counter is updated.

**Independent trigger with different LFSR generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). Then the LFSR2 counter is updated.
**Independent trigger with single triangle generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The DAC channel2 triangle counter is then updated.

**Independent trigger with different triangle generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The DAC channel2 triangle counter is then updated.

**Simultaneous software start**

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

In this configuration, one dac_pclk clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

**Simultaneous trigger without wave generation**

To configure the DAC in this conversion mode, the following sequence is required:
1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three dac_pclk clock cycles).

**Simultaneous trigger with single LFSR generation**

1. To configure the DAC in this conversion mode, the following sequence is required:

2. Set the two DAC channel trigger enable bits TEN1 and TEN2.

3. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.

4. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.

5. Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The LFSR2 counter is then updated.

**Simultaneous trigger with different LFSR generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2

2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.

3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits.

4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The LFSR1 counter is then updated.

At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The LFSR2 counter is then updated.

**Simultaneous trigger with single triangle generation**

To configure the DAC in this conversion mode, the following sequence is required:
1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The DAC channel2 triangle counter is then updated.

**Simultaneous trigger with different triangle generation**

To configure the DAC in this conversion mode, the following sequence is required:
1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). Then the DAC channel2 triangle counter is updated.

### 29.5 DAC low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSleep</td>
<td>No effect, the DAC can be used with DMA</td>
</tr>
<tr>
<td>CStop</td>
<td>The DAC remains active with a static output value. The Sample and hold mode is not available.</td>
</tr>
<tr>
<td>Standby</td>
<td>The DAC peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>
Table 218. Effect of low-power modes on DAC2

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSleep</td>
<td>No effect, the DAC can be used with DMA.</td>
</tr>
<tr>
<td>CStop</td>
<td>The DAC remains active with a static output value in Normal mode. The Sample and hold mode is available by using the LSI clock. In Autonomous mode (system Run and CStop), the DAC can stay active and can be used with DMA.</td>
</tr>
<tr>
<td>Standby</td>
<td>The DAC peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

29.6 DAC interrupts

Table 219. DAC interrupts

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit Sleep mode</th>
<th>Exit Stop mode</th>
<th>Exit Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAC</td>
<td>DMA underrun</td>
<td>DMAUDRx</td>
<td>DMAUDRI</td>
<td>Write DMAUDRx = 1</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
### 29.7 DAC registers

Refer to *Section 1 on page 97* for a list of abbreviations used in register descriptions. The peripheral registers have to be accessed by words (32-bit).

#### 29.7.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

| Bit 31 | Reserved, must be kept at reset value. |
| Bit 30 | CEN2: DAC channel2 calibration enable |
|        | This bit is set and cleared by software to enable/disable DAC channel2 calibration, it can be written only if EN2 bit is set to 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored. |
|        | 0: DAC channel2 in Normal operating mode |
|        | 1: DAC channel2 in calibration mode |
|        | *Note:* This bit is available only on dual-channel DACs. Refer to *Section 29.3: DAC implementation.* |
| Bit 29 | DMAUDRIE2: DAC channel2 DMA underrun interrupt enable |
|        | This bit is set and cleared by software. |
|        | 0: DAC channel2 DMA underrun interrupt disabled |
|        | 1: DAC channel2 DMA underrun interrupt enabled |
|        | *Note:* This bit is available only on dual-channel DACs. Refer to *Section 29.3: DAC implementation.* |
| Bit 28 | DMAEN2: DAC channel2 DMA enable |
|        | This bit is set and cleared by software. |
|        | 0: DAC channel2 DMA mode disabled |
|        | 1: DAC channel2 DMA mode enabled |
|        | *Note:* This bit is available only on dual-channel DACs. Refer to *Section 29.3: DAC implementation.* |

<table>
<thead>
<tr>
<th>Address offset: 0x00</th>
<th>Reset value: 0x0000 0000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>Bit 15-14</td>
<td>CEN1 DMAUDRIE1 DMAEN1 MAMP[3:0] WAVE[1:0] TSEL1[3:0] TSEL1[1:0] TEN1 EN1</td>
</tr>
<tr>
<td></td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>
Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

*Note:* These bits are available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable

These bits are set/reset by software.

00: wave generation disabled
01: Noise wave generation enabled
1x: Triangle wave generation enabled

*Note:* Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)

These bits are available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.

Bits 21:18 **TSEL2[3:0]**: DAC channel2 trigger selection

These bits select the external event used to trigger DAC channel2

0000: SWTRIG2
0001: dac_ch2_trig1
0010: dac_ch2_trig2
...
1111: dac_ch2_trig15

Refer to the trigger selection tables in Section 29.4.2: DAC pins and internal signals for details on trigger configuration and mapping.

*Note:* Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).

These bits are available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.

Bit 17 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

0: DAC channel2 trigger disabled and data written into the DAC_DHR2 register are transferred one dac_pclk clock cycle later to the DAC_DOR2 register
1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three dac_pclk clock cycles later to the DAC_DOR2 register

*Note:* When software trigger is selected, the transfer from the DAC_DHR2 register to the DAC_DOR2 register takes only one dac_pclk clock cycle.

These bits are available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.
Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.

- 0: DAC channel2 disabled
- 1: DAC channel2 enabled

*Note: These bits are available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.*

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CEN1**: DAC channel1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1=0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

- 0: DAC channel1 in Normal operating mode
- 1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

- 0: DAC channel1 DMA Underrun Interrupt disabled
- 1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

- 0: DAC channel1 DMA mode disabled
- 1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

- 0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
- 0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
- 0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
- 0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
- 0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
- 0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
- 0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
- 0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
- 1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
- 1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
- 1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
- ≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.

- 00: wave generation disabled
- 01: Noise wave generation enabled
- 1x: Triangle wave generation enabled

Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).
Bits 5:2 **TSEL[3:0]**: DAC channel1 trigger selection
These bits select the external event used to trigger DAC channel1
0000: SWTRIG1
0001: dac_ch1_trig1
0010: dac_ch1_trig2
...
1111: dac_ch1_trig15
Refer to the trigger selection tables in **Section 29.4.2: DAC pins and internal signals** for details on trigger configuration and mapping.
*Note: Only used if bit **TEN1** = 1 (DAC channel1 trigger enabled).*

Bit 1 **TEN1**: DAC channel1 trigger enable
This bit is set and cleared by software to enable/disable DAC channel1 trigger.
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_pclk clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_pclk clock cycles later to the DAC_DOR1 register
*Note: When software trigger is selected, the transfer from the DAC_DHR1 register to the DAC_DOR1 register takes only one dac_pclk clock cycle.*

Bit 0 **EN1**: DAC channel1 enable
This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

### 29.7.2 DAC software trigger register (DAC_SWTRGR)

Address offset: 0x04
Reset value: 0x0000 0000

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

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

**Bit 1 SWTRIG2**: DAC channel2 software trigger
This bit is set by software to trigger the DAC in software trigger mode.
0: No trigger
1: Trigger

*Note: This bit is cleared by hardware (one dac_pclk clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.
This bit is available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.*

**Bit 0 SWTRIG1**: DAC channel1 software trigger
This bit is set by software to trigger the DAC in software trigger mode.
0: No trigger
1: Trigger

*Note: This bit is cleared by hardware (one dac_pclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.*

### 29.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08
Reset value: 0x0000 0000

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

**Bits 11:0** DACC1DHR[11:0]: DAC channel1 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel1.

### 29.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C
Reset value: 0x0000 0000

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

**Bits 11:0** DACC1DHR[11:0]: DAC channel1 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel1.
29.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10
Reset value: 0x0000 0000

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 DACC1DHR[11:0]: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

29.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

This register is available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.

Address offset: 0x14
Reset value: 0x0000 0000

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

Bits 7:0 DACC1DHR[7:0]: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.
29.7.7 **DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)**

This register is available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.

Address offset: 0x18

Reset value: 0x0000 0000

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

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:4 **DAC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0  Reserved, must be kept at reset value.

29.7.8 **DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)**

This register is available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.

Address offset: 0x1C

Reset value: 0x0000 0000

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

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

Bits 7:0 **DAC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.
29.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20
Reset value: 0x0000 0000

| Bits 31:28 Reserved, must be kept at reset value. |
| Bits 27:16 **DAC2DHR[11:0]**: DAC channel2 12-bit right-aligned data |
| These bits are written by software which specifies 12-bit data for DAC channel2. |
| Bits 15:12 Reserved, must be kept at reset value. |
| Bits 11:0 **DAC1DHR[11:0]**: DAC channel1 12-bit right-aligned data |
| These bits are written by software which specifies 12-bit data for DAC channel1. |

29.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24
Reset value: 0x0000 0000

| Bits 31:20 **DAC2DHR[11:0]**: DAC channel2 12-bit left-aligned data |
| These bits are written by software which specifies 12-bit data for DAC channel2. |
| Bits 19:16 Reserved, must be kept at reset value. |
| Bits 15:4 **DAC1DHR[11:0]**: DAC channel1 12-bit left-aligned data |
| These bits are written by software which specifies 12-bit data for DAC channel1. |
| Bits 3:0 Reserved, must be kept at reset value. |
29.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28
Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DAC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DAC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

29.7.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C
Reset value: 0x0000 0000

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DAC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.
### 29.7.13 DAC channel2 data output register (DAC_DOR2)

This register is available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.

**Address offset:** 0x30

**Reset value:** 0x0000 0000

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

**DACC2DOR[11:0]**

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DOR[11:0]:** DAC channel2 data output

These bits are read-only, they contain data output for DAC channel2.

### 29.7.14 DAC status register (DAC_SR)

**Address offset:** 0x34

**Reset value:** 0x0000 0000

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

**BWST2**

CAL  | DMAU  | DR2  |
FLAG2|       |      |
     |       |      |
| r  | r  | rc_w1 |

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>rc_w1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**BWST1**

CAL  | DMAU  | DR1  |
FLAG1|       |      |
| r  | r  | rc_w1 |
Bit 31  **BWST2**: DAC channel2 busy writing sample time flag
This bit is systematically set just after Sample and hold mode enable. It is set each time the software writes the register DAC_SHSR2. It is cleared by hardware when the write operation of DAC_SHSR2 is complete. (It takes about 3 LSI periods of synchronization).
0: There is no write operation of DAC_SHSR2 ongoing: DAC_SHSR2 can be written
1: There is a write operation of DAC_SHSR2 ongoing: DAC_SHSR2 cannot be written
*Note:* This bit is available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.

Bit 30  **CAL_FLAG2**: DAC channel2 calibration offset status
This bit is set and cleared by hardware
0: calibration trimming value is lower than the offset correction value
1: calibration trimming value is equal or greater than the offset correction value
*Note:* This bit is available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.

Bit 29  **DMAUDR2**: DAC channel2 DMA underrun flag
This bit is set by hardware and cleared by software (by writing it to 1).
0: No DMA underrun error condition occurred for DAC channel2
1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).
*Note:* This bit is available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.

Bit 28  Reserved, must be kept at reset value.

Bit 27  Reserved, must be kept at reset value.

Bits 26:16  Reserved, must be kept at reset value.

Bit 15  **BWST1**: DAC channel1 busy writing sample time flag
This bit is systematically set just after Sample and hold mode enable and is set each time the software writes the register DAC_SHSR1. It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI periods of synchronization).
0: There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written
1: There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14  **CAL_FLAG1**: DAC channel1 calibration offset status
This bit is set and cleared by hardware
0: calibration trimming value is lower than the offset correction value
1: calibration trimming value is equal or greater than the offset correction value

Bit 13  **DMAUDR1**: DAC channel1 DMA underrun flag
This bit is set by hardware and cleared by software (by writing it to 1).
0: No DMA underrun error condition occurred for DAC channel1
1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12  Reserved, must be kept at reset value.

Bit 11  Reserved, must be kept at reset value.

Bits 10:0  Reserved, must be kept at reset value.
### 29.7.15 DAC calibration control register (DAC_CCR)

Address offset: 0x38  
Reset value: 0x00XX 00XX

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

Bits 31:21  Reserved, must be kept at reset value.

Bits 20:16  OTRIM2[4:0]: DAC channel2 offset trimming value

These bits are available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.

Bits 15:5  Reserved, must be kept at reset value.

Bits 4:0  OTRIM1[4:0]: DAC channel1 offset trimming value

### 29.7.16 DAC mode control register (DAC_MCR)

Address offset: 0x3C  
Reset value: 0x0000 0000

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

Bits 31:26  Reserved, must be kept at reset value.

Bit 25  Reserved, must be kept at reset value.

Bit 24  Reserved, must be kept at reset value.

Bits 23:19  Reserved, must be kept at reset value.
Bits 18:16 **MODE2[2:0]**: DAC channel2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2=0 and bit CEN2 =0 in the DAC_CR register). If EN2=1 or CEN2 =1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel2 mode:

- DAC channel2 in Normal mode
  - 000: DAC channel2 is connected to external pin with Buffer enabled
  - 001: DAC channel2 is connected to external pin and to on chip peripherals with buffer enabled
  - 010: DAC channel2 is connected to external pin with buffer disabled
  - 011: DAC channel2 is connected to on chip peripherals with Buffer disabled

- DAC channel2 in Sample and hold mode
  - 100: DAC channel2 is connected to external pin with Buffer enabled
  - 101: DAC channel2 is connected to external pin and to on chip peripherals with Buffer enabled
  - 110: DAC channel2 is connected to external pin and to on chip peripherals with Buffer disabled
  - 111: DAC channel2 is connected to on chip peripherals with Buffer disabled

*Note:* This register can be modified only when EN2=0.

Refer to Section 29.3: DAC implementation for the availability of DAC channel2.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

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

Bits 2:0 **MODE1[2:0]**: DAC channel1 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1=0 and bit CEN1 =0 in the DAC_CR register). If EN1=1 or CEN1 =1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel1 mode:

- DAC channel1 in Normal mode
  - 000: DAC channel1 is connected to external pin with Buffer enabled
  - 001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
  - 010: DAC channel1 is connected to external pin with Buffer disabled
  - 011: DAC channel1 is connected to on chip peripherals with Buffer disabled

- DAC channel1 in sample & hold mode
  - 100: DAC channel1 is connected to external pin with Buffer enabled
  - 101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
  - 110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled
  - 111: DAC channel1 is connected to on chip peripherals with Buffer disabled

*Note:* This register can be modified only when EN1=0.
29.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1)

Address offset: 0x40
Reset value: 0x0000 0000

| Bits 31:10 | Reserved, must be kept at reset value. |
| Bits 9:0   | TSAMPLE1[9:0]: DAC channel1 sample time (only valid in Sample and hold mode) |
These bits can be written when the DAC channel1 is disabled or also during normal operation. In the latter case, the write can be done only when BWST1 of DAC_SR register is low. If BWST1=1, the write operation is ignored.

Note: It represents the number of LSI clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x LSI clock period.

29.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2)

This register is available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.
Address offset: 0x44
Reset value: 0x0000 0000

| Bits 31:10 | Reserved, must be kept at reset value. |
| Bits 9:0   | TSAMPLE2[9:0]: DAC channel2 sample time (only valid in Sample and hold mode) |
These bits can be written when the DAC channel2 is disabled or also during normal operation. In the latter case, the write can be done only when BWST2 of DAC_SR register is low. If BWST2=1, the write operation is ignored.

Note: It represents the number of LSI clocks to perform a sample phase. Sampling time = (TSAMPLE2[9:0] + 1) x LSI clock period.
29.7.19 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48
Reset value: 0x0001 0001

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 THOLD2[9:0]: DAC channel 2 hold time (only valid in Sample and hold mode).

Hold time = (THOLD[9:0]) x LSI clock period

*Note:* This register can be modified only when EN2=0.

These bits are available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 THOLD1[9:0]: DAC channel 1 hold time (only valid in Sample and hold mode)

Hold time = (THOLD[9:0]) x LSI clock period

*Note:* This register can be modified only when EN1=0.

*Note:* These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit ENx=0 and bit CENx=0 in the DAC_CR register). If ENx=1 or CENx=1 the write operation is ignored.

29.7.20 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C
Reset value: 0x0001 0001

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:18 TREFRESH2[7:0]: DAC channel 2 refresh time (only valid in Sample and hold mode)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>-----------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Digital-to-analog converter (DAC)

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:16  \text{TREFRESH2}[7:0]: DAC channel2 refresh time (only valid in Sample and hold mode)
           Refresh time = \((\text{TREFRESH2}[7:0]) \times \text{LSI} \) clock period
           \textbf{Note:}  This register can be modified only when EN2=0.
           These bits are available only on dual-channel DACs. Refer to Section 29.3: DAC implementation.

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:0  \text{TREFRESH1}[7:0]: DAC channel1 refresh time (only valid in Sample and hold mode)
           Refresh time = \((\text{TREFRESH1}[7:0]) \times \text{LSI} \) clock period
           \textbf{Note:}  This register can be modified only when EN1=0.

\textbf{Note:}  These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit ENx=0 and bit CENx=0 in the DAC_CR register). If ENx=1 or CENx=1 the write operation is ignored.
## 29.7.21 DAC register map

Table 220 summarizes the DAC registers.

### Table 220: DAC register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | DAC_CR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x04   | DAC_SWTRGR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x08   | DAC_DHR12R1   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0C   | DAC_DHR12L1   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x10   | DAC_DHR8R1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x14   | DAC_DHR12R2   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x18   | DAC_DHR12L2   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1C   | DAC_DHR8R2    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x20   | DAC_DHR12RD   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x24   | DAC_DHR12LD   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x28   | DAC_DHR8RD    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x2C   | DAC_DOR1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x30   | DAC_DOR2      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x34   | DAC_SR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
Table 220. DAC register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x38   | DAC_CCR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x3C   | DAC_MCR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x40   | DAC_SHSR1     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x44   | DAC_SHSR2     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x48   | DAC_SHHR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4C   | DAC_SHHR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.3 on page 128 for the register boundary addresses.
30 Voltage reference buffer (VREFBUF)

30.1 Introduction

The devices embed a voltage reference buffer which can be used as voltage reference for ADCs, DACs and also as voltage reference for external components through the VREF+ pin.

30.2 VREFBUF functional description

The internal voltage reference buffer supports four voltages\(^{(a)}\), which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 000: around 2.5 V.
- VRS = 001: around 2.048 V.
- VRS = 010: around 1.8 V.
- VRS = 011: around 1.5 V.

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

<table>
<thead>
<tr>
<th>ENVR</th>
<th>HIZ</th>
<th>VREF buffer configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>VREFBUF buffer OFF:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREF+ pin pulled-down to VSSA</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>External voltage reference mode (default value):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREFBUF buffer OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREF+ pin input mode</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Internal voltage reference mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREFBUF buffer ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREF+ pin connected to VREFBUF buffer output</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Hold mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREFBUF buffer OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREF+ pin floating. The voltage is held with the external capacitor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VRR detection disabled and VRR bit keeps last state</td>
</tr>
</tbody>
</table>

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

---

\(^{(a)}\) The minimum \(V_{DDA}\) voltage depends on VRS setting, refer to the product datasheet.
### 30.3 VREFBUF registers

#### 30.3.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00  
Reset value: 0x0000 0002

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

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **VRS[2:0]**: Voltage reference scale  
These bits select the value generated by the voltage reference buffer.  
000: Voltage reference set to 2.5 V  
001: Voltage reference set to 2.048 V  
010: Voltage reference set to 1.8 V  
011: Voltage reference set to 1.5 V  
Others: Reserved

Bit 3 **VRR**: Voltage reference buffer ready  
0: the voltage reference buffer output is not ready.  
1: the voltage reference buffer output reached the requested level.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HIZ**: High impedance mode  
This bit controls the analog switch to connect or not the VREF+ pin.  
0: VREF+ pin is internally connected to the voltage reference buffer output.  
1: VREF+ pin is high impedance.  
Refer to Table 221: VREF buffer modes for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR**: Voltage reference buffer mode enable  
This bit is used to enable the voltage reference buffer mode.  
0: Internal voltage reference mode disable (external voltage reference mode).  
1: Internal voltage reference mode (reference buffer enable or hold mode) enable.
30.3.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04
Reset value: 0x0000 00XX

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

Bits 31:6  Reserved, must be kept at reset value.

Bits 5:0  TRIM[5:0]: Trimming code
These bits are automatically initialized after reset with the trimming value stored in the Flash memory during the production test. Writing into these bits allows the tuning of the internal reference buffer voltage.

30.3.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
31 Comparator (COMP)

31.1 Introduction

The device embeds two ultra-low-power comparator channels (COMP1 and COMP2). They can be used for a variety of functions including:

- wake up from low-power mode triggered by an analog signal
- analog signal conditioning
- cycle-by-cycle current control loop when combined with a PWM output from a timer

31.2 COMP main features

- Selectable inverting analog inputs:
  - Two I/O pins (different for either channel)
  - DAC1 channel 1 and channel 2 outputs
  - DAC2 channel 1 output
  - Internal reference voltage and three sub multiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
  - \( V_{\text{BAT/4}} \) and \( V_{\text{DDA}} \) supplies
  - Temperature monitoring
- Selectable as non-inverting analog inputs:
  - Two I/O pins per channel (different for either channel)
  - DAC2 channel 1 output
- Programmable hysteresis
- Programmable speed / consumption
- Mapping of outputs to I/Os
- Redirection of outputs to timer inputs for triggering:
  - capture events
  - OCREF_CLR events (for cycle-by-cycle current control)
  - break events for fast PWM shut downs
- Blanking of comparator outputs
- Window comparator
- Interrupt generation capability with wake up from Sleep and Stop modes (through the EXTI controller)
- Direct interrupt output to the CPU
31.3  COMP functional description

31.3.1  COMP block diagram

The block diagram of the comparators is shown in Figure 236: Comparator block diagram.

Figure 236. Comparator block diagram
### 31.3.2 COMP pins and internal signals

The I/Os used as comparator inputs must be configured in analog mode in the GPIO registers.

The comparator outputs can be connected to the I/Os through their alternate functions. Refer to the product datasheet.

The outputs can also be internally redirected to a variety of timer inputs for the following purposes:

- emergency shut-down of PWM signals, using BKIN and BKIN2 inputs
- cycle-by-cycle current control, using ETR inputs of timers
- input capture for timing measurements

The comparator output can be routed simultaneously internally and to the I/O pins.

---

#### Table 223. COMP input/output internal signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>compx_inm[4:9]</td>
<td>Analog input</td>
<td>Inverting inputs for COMPx (x = 1 or 2)</td>
</tr>
<tr>
<td>comp1_inp[1:3]</td>
<td>Analog input</td>
<td>Non-inverting inputs for COMP1</td>
</tr>
<tr>
<td>comp2_inp[1:2]</td>
<td>Analog input</td>
<td>Non-inverting inputs for COMP2</td>
</tr>
<tr>
<td>VREFINT(1)</td>
<td>Analog input</td>
<td>Internal reference voltage</td>
</tr>
<tr>
<td>comp_blk1</td>
<td>Digital input</td>
<td>Blanking input source for both COMP channels: TIM1 OC5</td>
</tr>
<tr>
<td>comp_blk2</td>
<td>Digital input</td>
<td>Blanking input source for both COMP channels: TIM2 OC3</td>
</tr>
<tr>
<td>comp_blk3</td>
<td>Digital input</td>
<td>Blanking input source for both COMP channels: TIM3 OC3</td>
</tr>
<tr>
<td>comp_blk4</td>
<td>Digital input</td>
<td>Blanking input source for both COMP channels: TIM3 OC4</td>
</tr>
<tr>
<td>comp_blk5</td>
<td>Digital input</td>
<td>Blanking input source for both COMP channels: TIM8 OC5</td>
</tr>
<tr>
<td>comp_blk6</td>
<td>Digital input</td>
<td>Blanking input source for both COMP channels: TIM15 OC1</td>
</tr>
<tr>
<td>comp_pclk</td>
<td>Digital input</td>
<td>APB clock for both COMP channels</td>
</tr>
<tr>
<td>comp1_wkup</td>
<td>Digital output</td>
<td>COMP1 channel 1 wakeup</td>
</tr>
<tr>
<td>comp1_out</td>
<td>Digital output</td>
<td>COMP1 channel 1 output</td>
</tr>
<tr>
<td>comp2_wkup</td>
<td>Digital output</td>
<td>COMP2 channel 2 wakeup</td>
</tr>
<tr>
<td>comp2_out</td>
<td>Digital output</td>
<td>COMP2 channel 2 output</td>
</tr>
<tr>
<td>comp_it</td>
<td>Digital output</td>
<td>COMP interrupt</td>
</tr>
</tbody>
</table>

1. 1/4V\text{REF\_COMP}, 1/2V\text{REF\_COMP}, 3/4V\text{REF\_COMP} and \text{V\text{REF\_COMP}} voltage levels are generated from \text{V\text{REF\_INT}}.
31.3.3 COMP reset and clocks

The clock comp_pclk provided by the clock controller is synchronous with the APB clock.

**Note:** *Important: The polarity selection logic and the output redirection to the port works independently from the APB clock. This allows the comparator to work even in Stop mode. The interrupt line, connected to the NVIC of CPU, requires the APB clock (comp_pclk) to work. In absence of the APB clock, the interrupt signal comp_it cannot be generated.*

31.3.4 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications with specific functional safety requirements, the comparator

### Table 224. COMP input/output pins

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPx_INP[1:2]</td>
<td>Analog input</td>
<td>Inverting inputs for COMPx (x = 1 or 2)</td>
</tr>
<tr>
<td>COMP1_INM[6:7]</td>
<td>Analog input</td>
<td>Non-inverting inputs for COMP1</td>
</tr>
<tr>
<td>COMP2_INM[6:7,9]</td>
<td>Analog input</td>
<td>Non-inverting inputs for COMP2</td>
</tr>
<tr>
<td>COMP1_OUT</td>
<td>Digital output</td>
<td>COMP channel 1 output: see Section 31.3.8: Comparator output on GPIOs.</td>
</tr>
<tr>
<td>COMP2_OUT</td>
<td>Digital output</td>
<td>COMP channel 2 output: see Section 31.3.8: Comparator output on GPIOs.</td>
</tr>
</tbody>
</table>

### Table 225. COMP interconnection

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREFINT</td>
<td>1/4VREF_comp (COMP1 and COMP2)</td>
</tr>
<tr>
<td></td>
<td>1/2VREF_comp (COMP1 and COMP2)</td>
</tr>
<tr>
<td></td>
<td>3/4VREF_comp (COMP1 and COMP2)</td>
</tr>
<tr>
<td></td>
<td>VREF_comp (COMP1 and COMP2)</td>
</tr>
<tr>
<td>comp1_inm4</td>
<td>DAC1 channel 1 (COMP1 and COMP2)</td>
</tr>
<tr>
<td>comp1_inm5</td>
<td>DAC1 channel 2 (COMP1 and COMP2)</td>
</tr>
<tr>
<td>comp1_inm6</td>
<td>PB1</td>
</tr>
<tr>
<td>comp2_inm6</td>
<td>PE10</td>
</tr>
<tr>
<td>comp1_inm7</td>
<td>PC4</td>
</tr>
<tr>
<td>comp2_inm7</td>
<td>PE7</td>
</tr>
<tr>
<td>comp1_inm8</td>
<td>Temperature monitoring (COMP1)</td>
</tr>
<tr>
<td>comp2_inm8</td>
<td>DAC2 channel 1 (COMP2)</td>
</tr>
<tr>
<td>comp1_inm9</td>
<td>1.4VBAT (COMP1)</td>
</tr>
<tr>
<td>comp2_inm9</td>
<td>VDDA supply voltage (COMP2)</td>
</tr>
<tr>
<td>comp1_inp3</td>
<td>DAC2 channel 1 (COMP1)</td>
</tr>
</tbody>
</table>
configuration can be protected against undesired alteration that could happen, for example, at program counter corruption.

For this purpose, the comparator configuration registers can be write-protected (read-only). Upon configuring a comparator channel, its LOCK bit is set to 1. This causes the whole register set of the comparator channel, as well as the common COMP_OR register, to become read-only, the LOCK bit inclusive.

The write protection can only be removed through the MCU reset. The COMP_OR register is locked by the LOCK bit of COMP_CFGR1 OR COMP_CFGR2.

31.3.5 Window comparator

The purpose of the window comparator is to monitor the analog voltage and check that it is comprised within the specified voltage range defined by lower and upper thresholds.

The window comparator requires both COMP channels. The monitored analog voltage is connected to their non-inverting (plus) inputs and the upper and lower threshold voltages are connected to the inverting (minus) input of either comparator, respectively. The non-inverting input of the COMP channel 2 can be connected internally with the non-inverting input of the COMP channel 1 by enabling WINMODE bit. This can save the input pins of COMP channel 2 for other purposes. See Figure 236: Comparator block diagram.

31.3.6 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed (for instance when exiting from low-power mode) to be able to force the hysteresis value using external components.

![Figure 237. Comparator hysteresis](image)

31.3.7 Comparator output blanking function

The purpose of the blanking function is to prevent the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches anti parallel diodes). It uses a blanking window defined with a timer output compare signal. Refer to the register description for selectable blanking signals. The
blanking signal gates the internal comparator output such as to clean the comp_out from spurious pulses due to current spikes, as depicted in Figure 238 (the COMP channel number is not represented).

**Figure 238. Comparator output blanking**

![Comparator output blanking diagram](image)

### 31.3.8 Comparator output on GPIOs

The COMP1_OUT and COMP2_OUT outputs of the comparator channels are mapped to GPIOs through the AFOP field of the COMP_OR register, bits [10:0], and through the GPIO alternate function.

<table>
<thead>
<tr>
<th>COMP1_OUT</th>
<th>Alternate Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC5</td>
<td>AF13</td>
</tr>
<tr>
<td>PE12</td>
<td>AF13</td>
</tr>
<tr>
<td>PA6</td>
<td>AF10, AF12 (can be used as timer break in)</td>
</tr>
<tr>
<td>PA8</td>
<td>AF12 (can be used as timer break in)</td>
</tr>
<tr>
<td>PB12</td>
<td>AF13 (can be used as timer break in)</td>
</tr>
<tr>
<td>PE6</td>
<td>AF11 (can be used as timer break in)</td>
</tr>
<tr>
<td>PE15</td>
<td>AF13 (can be used as timer break in)</td>
</tr>
<tr>
<td>PG2</td>
<td>AF11 (can be used as timer break in)</td>
</tr>
<tr>
<td>PG3</td>
<td>AF11 (can be used as timer break in)</td>
</tr>
<tr>
<td>PG4</td>
<td>AF11 (can be used as timer break in)</td>
</tr>
</tbody>
</table>
The assignment to GPIOs for both comparator channel outputs must be done before locking registers of any channel, because the common COMP.OR register is locked when locking the registers of either comparator channel.

### 31.3.9 Comparator output redirection

The outputs of either COMP channel can be redirected to timer break inputs (TIMx.BKIN or TIMx.BKIN2), as shown in Figure 239. For that end, the COMP channel output is connected to one of GPIOs programmable in alternate function as timer break input. See Table 226 and Table 227. The selected GPIO(s) must be set in open drain mode. The COMP output passes through the GPIO to the timer break input. With a pull-up resistor, the selected GPIO can be used as timer break input logic OR-ed with the comparator output.
31.3.10 COMP power and speed modes

The power consumption of the COMP channels versus propagation delay can be adjusted to have the optimum trade-off for a given application.

The bits PWRMODE[1:0] in COMP_CFGRx registers can be programmed as follows:

- 00: High speed / full power
- 01: Medium speed / medium power
- 10: Medium speed / medium power
- 11: Very-low speed / ultra-low-power
31.3.11 Scaler function

The scaler block is available to provide the different voltage reference levels to the comparator inputs. It is based on an amplifier driving a resistor bridge. The amplifier input is connected to the internal voltage reference.

The amplifier and the resistor bridge can be enabled separately. The amplifier is enabled by the SCALEN bits of the COMP_CFGRx registers. The resistor bridge is enabled by the BRGEN bits of the COMP_CFGRx registers.

When the resistor divided voltage is not used, the resistor bridge can be disconnected in order to reduce the consumption. When it is disconnected, the 1/4 $V_{\text{REF_COMP}}$, 1/2 $V_{\text{REF_COMP}}$ and 3/4 $V_{\text{REF_COMP}}$ levels are equal to $V_{\text{REF_COMP}}$.

![Figure 240. Scaler block diagram](image)

31.4 COMP low-power modes

Table 228. Comparator behavior in the low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>No effect on the comparators. Comparator interrupts cause the device to exit the Stop mode.</td>
</tr>
</tbody>
</table>

Note: The comparators cannot be used to exit the device from Sleep or Stop mode when the internal reference voltage is switched off.
31.5 COMP interrupts

There are two ways to use the comparator as interrupt source.

The comparator outputs are internally connected to the Extended interrupt and event controller. Each comparator has its own EXTI line and can generate either interrupts or events to make the device exit low-power modes.

The comparators also provide an interrupt line to the NVIC of CPU. This functionality is used when the CPU is active to handle low latency interrupt. It requires APB clock running.

31.5.1 Interrupt through EXTI block

Refer to Interrupt and events section for more details.

Sequence to enable the COMPx interrupt through EXTI block:
1. Configure the EXTI line, receiving the comp_wkup signal, in interrupt mode, select the rising, falling or either-edge sensitivity and enable the EXTI line.
2. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines.
3. Enable the COMPx.

Table 229. Interrupt control bits

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop modes</th>
<th>Exit from Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>comp1_wkup</td>
<td>through EXTI</td>
<td>through EXTI</td>
<td>yes</td>
<td>yes</td>
<td>N/A</td>
</tr>
<tr>
<td>comp2_whup</td>
<td>through EXTI</td>
<td>through EXTI</td>
<td>yes</td>
<td>yes</td>
<td>N/A</td>
</tr>
</tbody>
</table>

31.5.2 Interrupt through NVIC of the CPU

Sequence to enable the COMPx interrupt through NVIC of the CPU:
1. Configure and enable the NVIC IRQ channel mapped to the comp_it line.
2. Configure and enable the ITEN in COMP_CFGRx.
3. Enable the COMPx.

Table 230. Interrupt control bits

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Interrupt flag</th>
<th>Enable control bit</th>
<th>Interrupt clear bit</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>comp_it</td>
<td>C1IF in</td>
<td>ITEN in COMP_CFGR1</td>
<td>CC1IF</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>comp_it</td>
<td>C2IF in</td>
<td>ITEN in COMP_CFGR2</td>
<td>CC2IF</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Note: It is mandatory to enable APB clock to use this interrupt. If clock is not enabled, interrupt is not generated.
### COMP registers

#### 31.6.1 COMP status register (COMP_SR)

The COMP_SR is the comparator status register.

- **Address offset:** 0x00
- **System reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Address</th>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

- **Bits 31:18** Reserved, must be kept at reset value.

- **Bit 17** C2IF: COMP channel 2 Interrupt Flag
  - This bit is set by hardware when the COMP channel 2 output is set
  - This bit is cleared by software writing 1 the CC2IF bit in the COMP_ICFR register.

- **Bit 16** C1IF: COMP channel 1 Interrupt Flag
  - This bit is set by hardware when the COMP channel 1 output is set
  - This bit is cleared by software writing 1 the CC1IF bit in the COMP_ICFR register.

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

- **Bit 1** C2VAL: COMP channel 2 output status bit
  - This bit is read-only. It reflects the current COMP channel 2 output taking into account POLARITY and BLANKING bits effect.

- **Bit 0** C1VAL: COMP channel 1 output status bit
  - This bit is read-only. It reflects the current COMP channel 1 output taking into account POLARITY and BLANKING bits effect.

#### 31.6.2 COMP interrupt clear flag register (COMP_ICFR)

The COMP_ICFR is the Comparator interrupt clear flag register.

- **Address offset:** 0x04
- **System reset value:** 0x0000 0004

<table>
<thead>
<tr>
<th>Address</th>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

- **Bit 17** C2IF: COMP channel 2 Interrupt Flag
  - This bit is cleared by software writing 1 the CC2IF bit in the COMP_ICFR register.

- **Bit 16** C1IF: COMP channel 1 Interrupt Flag
  - This bit is cleared by software writing 1 the CC1IF bit in the COMP_ICFR register.

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

- **Bit 1** C2VAL: COMP channel 2 output status bit
  - This bit is read-only. It reflects the current COMP channel 2 output taking into account POLARITY and BLANKING bits effect.

- **Bit 0** C1VAL: COMP channel 1 output status bit
  - This bit is read-only. It reflects the current COMP channel 1 output taking into account POLARITY and BLANKING bits effect.
31.6.3 **COMP option register [alternate] (COMP_OR)**

The COMP_OR is the Comparator option register.

**Address offset:** 0x08  
**System reset value:** 0x0000 0000

When OR_CFG=0:

<table>
<thead>
<tr>
<th>Bit 31-17</th>
<th>Description</th>
<th>Bit 16-10</th>
<th>Description</th>
<th>Bits 9-0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>OR15</td>
<td>OR14</td>
<td>OR13</td>
<td>OR12</td>
<td>OR11</td>
<td>AFOP[10:0]</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

When OR_CFG=1:

<table>
<thead>
<tr>
<th>Bit 31-16</th>
<th>Description</th>
<th>Bit 15-11</th>
<th>Description</th>
<th>Bits 10-0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>OR31</td>
<td>OR30</td>
<td>OR29</td>
<td>OR28</td>
<td>OR27</td>
<td>OR26</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:11 **OR:** Option register (when OR_CFG=0)

**Bits 10:0** **AFOP[10:0]:** Selection of source for alternate function of output ports

Bits of this field are set and cleared by software (only if LOCK not set).

Output port (GPIO) correspondence:

<table>
<thead>
<tr>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK2</td>
<td>PI4</td>
<td>PI1</td>
<td>PG4</td>
<td>PG3</td>
<td>PG2</td>
<td>PE15</td>
<td>PE6</td>
<td>PB12</td>
<td>PA8</td>
<td>PA6</td>
</tr>
</tbody>
</table>

For each bit:

- 0: COMP1_OUT is selected for the alternate function of the corresponding GPIO
- 1: COMP2_OUT is selected for the alternate function of the corresponding GPIO

When OR_CFG=1:

<table>
<thead>
<tr>
<th>Bit 31-16</th>
<th>Description</th>
<th>Bit 15-11</th>
<th>Description</th>
<th>Bits 10-0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>OR31</td>
<td>OR30</td>
<td>OR29</td>
<td>OR28</td>
<td>OR27</td>
<td>OR26</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:11 **OR:** Option register (when OR_CFG=0)
31.6.4 COMP configuration register 1 (COMP_CFRG1)

The COMP_CFRG1 is the COMP channel 1 configuration register.

Address offset: 0x0C

System reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 LOCK: Lock bit
This bit is set by software and cleared by a hardware system reset. It locks the whole content of the COMP channel 1 configuration register COMP_CFRG1[31:0], and COMP_OR register
0: COMP_CFRG1[31:0] register is read/write
1: COMP_CFRG1[31:0] and COMP_OR registers are read-only

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:24 BLANKING[3:0]: COMP channel 1 blanking source selection bits
Bits of this field are set and cleared by software (only if LOCK not set). The field selects the input source for COMP channel 1 output blanking:
0000: No blanking
0001: comp_blk1
0010: comp_blk2
0011: comp_blk3
0100: comp_blk4
0101: comp_blk5
0110: comp_blk6
All other values: reserved

Bit 23 Reserved, must be kept at reset value.
Bit 22  **INP2SEL**: COMP channel 1 non-inverting input 2 selection bit
This bit is set and cleared by software (only if LOCK not set).
0: DAC2 channel 1 input disabled
1: DAC2 channel 1 input enabled

*Note:* To avoid conflicts between INPLUS and INPLUS2, COMP1_INP1 and COMP1_INP2 automatically become floating when INP2SEL is set.

Bit 21  Reserved, must be kept at reset value.

Bit 20  **INPSEL**: COMP channel 1 non-inverting input selection bit
This bit is set and cleared by software (only if LOCK not set).
0: COMP1_INP1 (PB0)
1: COMP1_INP2 (PB2)

Bits 19:16  **INMSEL[3:0]**: COMP channel 1 inverting input selection field
These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input minus of COMP channel 1.
0000 = 1/4 VREF_COMP
0001 = 1/2 VREF_COMP
0010 = 3/4 VREF_COMP
0011 = VREF_COMP
0100 = COMP1_INM4 (DAC1 channel 1)
0101 = COMP1_INM5 (DAC1 channel 2)
0110 = COMP1_INM6 (PB1)
0111 = COMP1_INM7 (PC4)
1000 = COMP1_INM8 (temperature monitoring)
1001 = COMP1_INM9 (V_BAT/4)
Other configurations: Reserved

Bits 15:14  Reserved, must be kept at reset value.

Bits 13:12  **PWRMODE[1:0]**: Power Mode of the COMP channel 1
These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the COMP channel 1.
00: High speed / full power
01: Medium speed / medium power
10: Medium speed / medium power
11: Ultra low power / ultra-low-power

Bits 11:10  Reserved, must be kept at reset value.

Bits 9:8  **HYST[1:0]**: COMP channel 1 hysteresis selection bits
These bits are set and cleared by software (only if LOCK not set). They select the Hysteresis voltage of the COMP channel 1.
00: No hysteresis
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 7  Reserved, must be kept at reset value.

Bit 6  **ITEN**: COMP channel 1 interrupt enable
This bit is set and cleared by software (only if LOCK not set). This bit enable the interrupt generation of the COMP channel 1.
0: Interrupt generation disabled for COMP channel 1
1: Interrupt generation enabled for COMP channel 1
31.6.5 COMP configuration register 2 (COMP_CFGR2)

The COMP_CFGR2 is the COMP channel 2 configuration register.

Address offset: 0x10

System reset value: 0x0000 0000

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

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 POLARITY: COMP channel 1 polarity selection bit
- This bit is set and cleared by software (only if LOCK not set). It inverts COMP channel 1 polarity.
  0: COMP channel 1 output is not inverted
  1: COMP channel 1 output is inverted

Bit 2 SCALEN: Voltage scaler enable bit
- This bit is set and cleared by software (only if LOCK not set). This bit enables the VREFINT scaler for the COMP channels.
  0: VREFINT scaler disabled (if SCALEN bit of COMP_CFGR2 register is also low)
  1: VREFINT scaler enabled

Bit 1 BRGEN: Scaler bridge enable
- This bit is set and cleared by software (only if LOCK not set). This bit enables the bridge of the scaler.
  0: Scaler resistor bridge disabled (if BRGEN bit of COMP_CFGR2 register is also low)
  1: Scaler resistor bridge enabled
If SCALEN is set and BRGEN is reset, all four scaler outputs provide the same level VREF_COMP (similar to VREFINT).
If SCALEN and BRGEN are set, the four scaler outputs provide VREF_COMP, 3/4 VREF_COMP, 1/2 VREF_COMP and 1/4 VREF_COMP levels, respectively.

Bit 0 EN: COMP channel 1 enable bit
- This bit is set and cleared by software (only if LOCK not set). It enables the COMP channel 1.
  0: Disable
  1: Enable
Bit 31  LOCK: Lock bit
This bit is set by software and cleared by a hardware system reset. It locks the whole content
of the COMP channel 2 configuration register COMP_CFRG2[31:0], and COMP_OR register
0: COMP_CFRG2[31:0] register is read/write
1: COMP_CFRG2[31:0] and COMP_OR registers are read-only
Bits 30:28  Reserved, must be kept at reset value.
Bits 27:24  BLANKING[3:0]: COMP channel 2 blanking source selection bits
These bits are set and cleared by software (only if LOCK not set). These bits select which
timer output controls the COMP channel 2 output blanking.
0000: No blanking
0001: TIM1 OC5 selected as blanking source
0010: TIM2 OC3 selected as blanking source
0011: TIM3 OC3 selected as blanking source
0100: TIM3 OC4 selected as blanking source
0101: TIM8 OC5 selected as blanking source
0110: TIM15 OC1 selected as blanking source
Other configurations: Reserved
Bits 23:21  Reserved, must be kept at reset value.
Bit 20  INPSEL: COMP channel 2 non-inverting input selection bit
This bit is set and cleared by software (only if LOCK not set).
0: COMP2_INP1 (PE9)
1: COMP2_INP2 (PE11)
Bits 19:16  INMSEL[3:0]: COMP channel 2 inverting input selection field
These bits are set and cleared by software (only if LOCK not set). They select which input is
connected to the input minus of COMP channel 2.
0000 = 1/4 VREF_COMP
0001 = 1/2 VREF_COMP
0010 = 3/4 VREF_COMP
0011 = VREF_COMP
0100 = COMP2_INM4 (DAC1 channel 1)
0101 = COMP2_INM5 (DAC1 channel 2
0110 = COMP2_INM6 (PE10)
0111 = COMP2_INM7 (PE7)
1000 = COMP2_INM8 (DAC2 channel 1)
1001 = COMP2_INM9 (VDDA)
Other configurations: Reserved
Bits 15:14  Reserved, must be kept at reset value.
Bits 13:12  PWRRMODE[1:0]: Power Mode of the COMP channel 2
These bits are set and cleared by software (only if LOCK not set). They control the
power/speed of the COMP channel 2.
00: High speed / full power
01: Medium speed / medium power
10: Medium speed / medium power
11: Ultra low power / ultra-low-power
Bits 11:10  Reserved, must be kept at reset value.
Bits 9:8 \textbf{HYST[1:0]}: COMP channel 2 hysteresis selection bits

These bits are set and cleared by software (only if LOCK not set). They select the hysteresis voltage of the COMP channel 2.

00: No hysteresis
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 7 Reserved, must be kept at reset value.

Bit 6 \textbf{ITEN}: COMP channel 2 interrupt enable

This bit is set and cleared by software (only if LOCK not set). This bit enable the interrupt generation of the COMP channel 2.

0: Interrupt generation disabled for COMP channel 2
1: Interrupt generation enabled for COMP channel 2

Bit 5 Reserved, must be kept at reset value.

Bit 4 \textbf{WINMODE}: Window comparator mode selection bit

This bit is set and cleared by software (only if LOCK not set). This bit selects the window mode of the comparators. If set, the non-inverting input of COMP channel 2 is connected to the non-inverting input of the COMP channel 1.

Depending on the bit value, the non-inverting input of COMP channel 2 is connected to:

0: COMP2\_INP input selector
1: Non-inverting input comp1\_inp of COMP channel 1

Bit 3 \textbf{POLARITY}: COMP channel 2 polarity selection bit

This bit is set and cleared by software (only if LOCK not set). It inverts COMP channel 2 polarity.

0: COMP channel 2 output is not inverted
1: COMP channel 2 output is inverted

Bit 2 \textbf{SCALEN}: Voltage scaler enable bit

This bit is set and cleared by software (only if LOCK not set). This bit enables the \(V_{\text{REFINT}}\) scaler for the COMP channels.

0: \(V_{\text{REFINT}}\) scaler disabled (if SCALEN bit of COMP\_CFGR1 register is also low)
1: \(V_{\text{REFINT}}\) scaler enabled

Bit 1 \textbf{BRGEN}: Scaler bridge enable

This bit is set and cleared by software (only if LOCK not set). This bit enables the bridge of the scaler.

0: Scaler resistor bridge disabled (if BRGEN bit of COMP\_CFGR1 register is also low)
1: Scaler resistor bridge enabled

If SCALEN is set and BRGEN is reset, all four scaler outputs provide the same level \(V_{\text{REF\_COMP}}\) (similar to \(V_{\text{REFINT}}\)).

If SCALEN and BRGEN are set, the four scaler outputs provide \(V_{\text{REF\_COMP}}\), \(3/4 V_{\text{REF\_COMP}}\), \(1/2 V_{\text{REF\_COMP}}\) and \(1/4 V_{\text{REF\_COMP}}\) levels, respectively.

Bit 0 \textbf{EN}: COMP channel 2 enable bit

This bit is set and cleared by software (only if LOCK not set). It enables the COMP channel 2.

0: Disable
1: Enable
### 31.6.6 COMP register map

The following table summarizes the comparator registers.

| Offset | Register name          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | COMP_SR                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | COMP_ICFR              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | COMP_OR (OR_CFG=0)     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | AFOP[10:0]             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | COMP_OR (OR_CFG=1)     | OR31 | OR30 | OR29 | OR28 | OR27 | OR26 | OR25 | OR24 | OR23 | OR22 | OR21 | OR20 | OR19 | OR18 | OR17 | OR16 | OR15 | OR14 | OR13 | OR12 | OR11 | OR10 | OR9  | OR8  | OR7  | OR6  | OR5  | OR4  | OR3  | OR2  | OR1  | OR0  |
|        | Reset value            | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | AFOP[10:0]             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
32 Operational amplifiers (OPAMP)

32.1 Introduction

The devices embed two operational amplifiers with two inputs and one output each. The three I/Os can be connected to the external pins, thus enabling any type of external interconnections. The operational amplifiers can be configured internally as a follower, as an amplifier with a non-inverting gain ranging from 2 to 16 or with inverting gain ranging from -1 to -15.

Refer to Section 32.3.3: Signal routing for detailed information on OPAMP input and output connection to internal peripherals.

32.2 OPAMP main features

- Rail-to-rail input and output voltage range
- Low input bias current (down to 1 nA)
- Low input offset voltage (1.5 mV after calibration, 10 mV with factory calibration)
- 7 MHz gain bandwidth
- High-speed mode to achieve a better slew rate

*Note:* Refer to the product datasheet for detailed OPAMP characteristics.

32.3 OPAMP functional description

The OPAMP has several modes.

Each OPAMP can be individually enabled, when disabled the output is high-impedance.

When enabled, it can be in calibration mode, all input and output of the OPAMP are then disconnected, or in functional mode.

There are two functional modes, the high-speed mode and the normal mode. In functional mode the inputs and output of the OPAMP are connected as described in Section 32.3.3: Signal routing.

32.3.1 OPAMP reset and clocks

The operational amplifier clock is necessary for accessing the registers. When the application does not need to have read or write access to those registers, the clock can be switched off using the peripheral clock enable register (see OPAMPEN bit in Section 8.7.43: RCC APB1 clock register (RCC_APB1LENR)).

The bit OPAEN enables and disables the OPAMP operation. The OPAMP registers configurations should be changed before enabling the OPAEN bit in order to avoid spurious effects on the output.

When the output of the operational amplifier is no more needed the operational amplifier can be disabled to save power. All the configurations previously set (including the calibration) are maintained while OPAMP is disabled.
32.3.2 Initial configuration

The default configuration of the operational amplifier is a functional mode where the three input/outputs are connected to external pins. In the default mode the operational amplifier uses the factory trimming values for its offset calibration. See electrical characteristics section of the datasheet for factory trimming conditions, usually the temperature is 30 °C and the voltage is 3 V. The trimming values can be adjusted, see Section 32.3.5: Calibration for changing the trimming values. The default configuration uses the normal mode, which provides the standard performance. The bit OPAHSM can be set in order to switch the operational amplifier to high-speed mode for a better slew rate. Both normal and high-speed mode characteristics are defined in Section: Electrical characteristics of the datasheet.

As soon as the OPAEN bit in OPAMPx_CSR register is set, the operational amplifier is functional. The two input pins and the output pin are connected as defined in Section 32.3.3: Signal routing and the default connection settings can be changed.

Note: The inputs and output pins must be configured in analog mode (default state) in the corresponding GPIOx_MODER register.

32.3.3 Signal routing

The routing for the operational amplifier pins is determined by OPAMPx_CSR register. The connections of the two operational amplifiers (OPAMP1 and OPAMP2) are described in the table below.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Pin</th>
<th>Internal</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPAMP1_VINM</td>
<td>PC5(INM0)</td>
<td>ADC1_INP7, ADC1_INM3, ADC1_INP8, ADC1_INM4, ADC2_INP7, ADC2_INM3, ADC2_INP8, ADC2_INM4, COMP1_OUT, OPAMP1_VOUT or PGA</td>
<td>This pin is controlled by bits PGA_GAIN and VM_SEL.</td>
</tr>
<tr>
<td></td>
<td>PA7(INM1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPAMP1_VINP</td>
<td>PB0</td>
<td>dac1_out1, ADC1_INM5, ADC1_INP9, ADC2_INM5, ADC2_INP9, COMP1_INP1</td>
<td>This pin is controlled by bit VP_SEL.</td>
</tr>
<tr>
<td>OPAMP1_VOUT</td>
<td>PC4</td>
<td>ADC1_INP4, ADC2_INP4, COMP1_INM7</td>
<td>The pin is connected when the OPAMP is enabled. The ADC input is controlled by ADC.</td>
</tr>
</tbody>
</table>
32.3.4 OPAMP modes

The operational amplifier inputs and outputs are all accessible on terminals. The amplifiers can be used in multiple configuration environments:

- Standalone mode (external gain setting mode)
- Follower configuration mode
- PGA modes

**Note:** The amplifier output pin is directly connected to the output pad to minimize the output impedance. When the amplifier is enabled, it cannot be used as a general purpose I/O, even if the amplifier is configured as a PGA and only connected to the internal channel.

The impedance of the signal must be maintained below a level which avoids the input leakage to create significant artifacts (due to a resistive drop in the source). Please refer to the electrical characteristics section in the datasheet for further details.

### Standalone mode (external gain setting mode)

The procedure to use the OPAMP in standalone mode is presented hereafter.

Starting from the default value of OPAMPx_CSR, and the default state of GPIOx_MODER, as soon as the OPAEN bit is set, the two input pins and the output pin are connected to the operational amplifier.

This default configuration uses the factory trimming values and operates in normal mode (highest performance). The behavior of the OPAMP can be changed as follows:

- OPAHSM can be set to “operational amplifier high-speed” mode in order to have high slew rate.
- USERTRIM can be set to modify the trimming values for input offsets.
Follower configuration mode

The procedure to use the OPAMP in follower mode is presented hereafter.

- configure VM_SEL bits as "opamp_out connected to OPAMPx_VINM input", 11
- configure VP_SEL bits as "GPIO connected to OPAMPx_VINP", 00
- As soon as the OPAEN bit is set, the voltage on pin OPAMPx_VINP is buffered to pin OPAMPx_VOUT.

Note: The pin corresponding to OPAMPx_VINM is free for another usage.

The signal on the OPAMP1 output is also seen as an ADC input. As a consequence, the OPAMP configured in follower mode can be used to perform impedance adaptation on input signals before feeding them to the ADC input, assuming the input signal frequency is compatible with the operational amplifier gain bandwidth specification.
Figure 242. Follower configuration

Always connected to OPAMP output
Programmable gain amplifier mode

The procedure to use the OPAMP as programmable gain amplifier is presented hereafter.

- configure VM_SEL bits as "Feedback resistor is connected to OPAMPx_VINM input", 10
- configure PGA_GAIN bits as "internal Gain 2, 4, 8 or 16", 0000 to 0011
- configure VP_SEL bits as "GPIO connected to OPAMPx_VINP", 00

As soon as the OPAEN bit is set, the voltage on pin OPAMPx_VINP is amplified by the selected gain and visible on pin OPAMPx_VOUT.

Note: To avoid saturation, the input voltage should stay below VDDA divided by the selected gain.

Figure 243. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used
Programmable gain amplifier mode with external filtering

The procedure to use the OPAMP to amplify the amplitude of an input signal, with an external filtering, is presented hereafter.

- configure VM_SEL bits as “Feedback resistor is connected to OPAMPx_VINM input”, 10
- configure PGA_GAIN bits as “internal Gain 2, 4, 8 or 16 with filtering on INM0”, 0100 to 0111
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”.

Any external connection on INM can be used in parallel with the internal PGA, for example a capacitor can be connected between opamp_out and INM for filtering purpose (see datasheet for the value of resistors used in the PGA resistor network).

Figure 244. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering

1. The gain depends on the cut-off frequency.
Programmable gain amplifier, non-inverting with external bias or inverting mode

The procedure to use the OPAMP to amplify the amplitude of an input signal with bias voltage for non-inverting mode or inverting mode.

- configure VM_SEL bits as "Feedback resistor is connected to OPAMPx_VINM input",
- configure PGA_GAIN bits as "Inverting gain=-1,-3,-7,-15/ Non-inverting gain =2,4,8,16 with INM0", 1000 to 1011
- configure VP_SEL bits as "GPIO connected to OPAMPx_VINP".

**Figure 245. PGA mode, non-inverting gain setting (x2/x4/x8/x16) or inverting gain setting (x-1/x-3/x-7/x-15)**

**Figure 246. Example configuration**
Programmable gain amplifier, non-inverting with external bias or inverting mode with filtering

The procedure to use the OPAMP to amplify the amplitude of an input signal with bias voltage for non-inverting mode or inverting mode with filtering

- configure VM_SEL bits as "Feedback resistor is connected to OPAMPx_VINM input", 10
- configure PGA_GAIN bits as "Inverting gain=-1,-3,-7,-15/ Non-inverting gain =2,4,8,16 with INM0, INM1 node for filtering", 1100 to 1111
- configure VP_SEL bits as "GPIO connected to OPAMPx_VINP".

Any external connection on VM1 can be used in parallel with the internal PGA, for example a capacitor can be connected between opamp_out and VM1 for filtering purpose (see datasheet for the value of resistors used in the PGA resistor network).

**Figure 247.** PGA mode, non-inverting gain setting (x2/x4/x8/x16) or inverting gain setting (x-1/x-3/x-7/x-15) with filtering

**Figure 248.** Example configuration
32.3.5 Calibration

The OPAMP interface continuously sends trimmed offset values to the operational amplifiers. At startup, the trimming values are initialized with the preset ‘factory’ trimming value.

Each operational amplifier can be trimmed by the user. Specific registers allow to have different trimming values for normal mode and for high-speed mode.

The aim of the calibration is to cancel as much as possible the OPAMP inputs offset voltage. The calibration circuitry allows to reduce the input offset voltage to less than +/-1.5 mV within stable voltage and temperature conditions.

For each operational amplifier and each mode two trimming value needs to be trimmed, one for N differential pair and one for P differential pair.

There are two registers for trimming the offsets for each operational amplifiers, one for normal mode (OPAMPx_OTR) and one high-speed mode (OPAMPx_HSOTR). Each register is composed of five bits for P differential pair trimming and five bits for N differential pair trimming. These are the ‘user’ values.

The user is able to switch from ‘factory’ values to ‘user’ trimmed values using the USERTRIM bit in the OPAMPx_CSR register. This bit is reset at startup and so the ‘factory’ value are applied by default to the OPAMP option registers.

User is liable to change the trimming values in calibration or in functional mode.

The offset trimming registers are typically configured after the calibration operation is initialized by setting bit CALON to 1. When CALON = 1 the inputs of the operational amplifier are disconnected from the functional environment.

- Setting CALSEL to 01 initializes the offset calibration for the P differential pair (low voltage reference used).
- Resetting CALSEL to 11 initializes the offset calibration for the N differential pair (high voltage reference used).

When CALON = 1, the bit CALOUT will reflect the influence of the trimming value selected by CALSEL and OPAHSM. The software should increment the TRIMOFFSETN bits in the OPAMP control register from 0x00 to the first value that causes the CALOUT bit to change from 1 to 0 in the OPAMP register. If the CALOUT bit is reset, the offset is calibrated correctly and the corresponding trimming value must be stored. The CALOUT flag needs up to 1 ms after the trimming value is changed to become steady (see tOFFTRIMmax delay specification in the electrical characteristics section of the datasheet).

\[ \text{Note: } \text{The closer the trimming value is to the optimum trimming value, the longer it takes to stabilize (with a maximum stabilization time remaining below 1 ms in any case).} \]

<table>
<thead>
<tr>
<th>Mode</th>
<th>Control bits</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OPAEN</td>
<td>OPAHSM</td>
</tr>
<tr>
<td>Normal operating mode</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>High-speed mode</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Power down</td>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>
Calibration procedure

Here are the steps to perform a full calibration of either one of the operational amplifiers:

1. Set the OPAEN bit in OPAMPx_CSR to 1 to enable the operational amplifier.
2. Set the USERSTRIM bit in the OPAMPx_CSR register to 1.
3. Choose a calibration mode (refer to Table 233: Operating modes and calibration). The steps 3 to 4 will have to be repeated 4 times. For the first iteration select
   - Normal mode and N differential pair
   The above calibration mode correspond to OPAHSM=0 and CALSEL=11 in the OPAMPx_CSR register.
4. Increment TRIMOFFSETN[4:0] in OPAMPx_OTR starting from 00000b until CALOUT changes to 0 in OPAMPx_CSR.

Note: Between the write to the OPAMPx_OTR register and the read of the CALOUT value, make sure to wait for the TOFFTRIM\text{max} delay specified in the electrical characteristics section of the datasheet, to get the correct CALOUT value.

The commutation means that the is correctly compensated and that the corresponding trim code must be saved in the OPAMPx_OTR register.

Repeat steps 3 to 4 for:
- Normal mode and P differential pair, CALSEL=01
- High-speed mode and N differential pair
- High-speed mode and P differential pair

If a mode is not used, it is not necessary to perform the corresponding calibration.

All operational amplifier can be calibrated at the same time.

Note: During the whole calibration phase the external connection of the operational amplifier output must not pull up or down currents higher than 500 µA.
### 32.4 OPAMP low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect.</td>
</tr>
<tr>
<td>D2 Stop</td>
<td>No effect, OPAMP registers content is kept.</td>
</tr>
<tr>
<td>Standby</td>
<td>The OPAMP registers are powered down and must be re-initialized after exiting Standby.</td>
</tr>
</tbody>
</table>

#### Table 234. Effect of low-power modes on the OPAMP

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect.</td>
</tr>
<tr>
<td>D2 Stop</td>
<td>No effect, OPAMP registers content is kept.</td>
</tr>
<tr>
<td>Standby</td>
<td>The OPAMP registers are powered down and must be re-initialized after exiting Standby.</td>
</tr>
</tbody>
</table>

### 32.5 OPAMP PGA gain

When OPAMP is configured as PGA mode, it can select the gain of x2, x4, x8, x16 for non-inverting mode and x-1, x-3, x-7, x-15 for inverting mode.

When OPAMP is configured as non-inverting mode, the Gain error can be refer to the product datasheet. When it is configured as inverting mode, Gain factor is defined not only the on chip feedback resistor but also the signal source output impedance. If signal source output impedance is not negligible compare to the input feedback resistance of PGA, it will create the gain error. Please refer to the PGA resistance value in the product datasheet.

### 32.6 OPAMP registers

The registers of this peripheral can only be accessed by-word (32-bit).

#### 32.6.1 OPAMP1 control/status register (OPAMP1_CSR)

Address: 0x00

Reset value: 0x0000 0000

| Bit 31 | RES | Bit 30 | CALOUT | Bit 29 | TSTREF | Bit 28 | RES | Bit 27 | RES | Bit 26 | RES | Bit 25 | RES | Bit 24 | RES | Bit 23 | RES | Bit 22 | RES | Bit 21 | RES | Bit 20 | RES | Bit 19 | RES | Bit 18 | RES | Bit 17 | RES | Bit 16 | RES |
|--------|-----|--------|--------|--------|--------|--------|-----|--------|-----|--------|-----|--------|-----|--------|-----|--------|-----|--------|-----|--------|-----|--------|-----|--------|-----|--------|-----|--------|-----|
|        | Res.|        |  CAL OUT |       |  TST REF |  CAL | RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |  RES |
| 0x00   | 0x00|        |        |        |        |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |

| Bit 31 | Reserved, must be kept at reset value.                                      |
| Bit 30 | **CALOUT**: Operational amplifier calibration output                        |
|        | OPAMP output status flag. During the calibration mode, OPAMP is used as comparator. |
|        | 0: Non-inverting < inverting                                                |
|        | 1: Non-inverting > inverting                                                |
| Bit 29 | **TSTREF**: OPAMP calibration reference voltage output control (reserved for test) |
|        | 0: INTVREF of OPAMP is not output                                          |
|        | 1: INTVREF of OPAMP is output                                              |
Bits 28:19 Reserved, must be kept at reset value.

Bit 18 **USERTRIM**: User trimming enable
This bit allows to switch from ‘factory’ AOP offset trimmed values to ‘user’ AOP offset trimmed values
This bit is active for both mode normal and high-power.
0: ‘factory’ trim code used
1: ‘user’ trim code used

Bits 17:14 **PGA_GAIN**: Operational amplifier Programmable amplifier gain value
0000: Non-inverting internal Gain 2, VREF- referenced
0001: Non-inverting internal Gain 4, VREF- referenced
0010: Non-inverting internal Gain 8, VREF- referenced
0011: Non-inverting internal Gain 16, VREF- referenced
0100: Non-inverting internal Gain 2 with filtering on INM0, VREF- referenced
0101: Non-inverting internal Gain 4 with filtering on INM0, VREF- referenced
0110: Non-inverting internal Gain 8 with filtering on INM0, VREF- referenced
0111: Non-inverting internal Gain 16 with filtering on INM0, VREF- referenced
1000: Inverting gain=-1/ Non-inverting gain =2 with INM0 node for input or bias
1001: Inverting gain=-3/ Non-inverting gain =4 with INM0 node for input or bias
1010: Inverting gain=-7/ Non-inverting gain =8 with INM0 node for input or bias
1011: Inverting gain=-15/ Non-inverting gain =16 with INM0 node for input or bias
1100: Inverting gain=-1/ Non-inverting gain =2 with INM0 node for input or bias, INM1 node for filtering
1101: Inverting gain=-3/ Non-inverting gain =4 with INM0 node for input or bias, INM1 node for filtering
1110: Inverting gain=-7/ Non-inverting gain =8 with INM0 node for input or bias, INM1 node for filtering
1111: Inverting gain=-15/ Non-inverting gain =16 with INM0 node for input or bias, INM1 node for filtering

Bits 13:12 **CALSEL**: Calibration selection
It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP= 1.
00: 0.033*VDDA applied on OPAMP inputs
01: 0.1*VDDA applied on OPAMP inputs (for PMOS calibration)
10: 0.5*VDDA applied on OPAMP inputs
11: 0.9*VDDA applied on OPAMP inputs (for NMOS calibration)

Bit 11 **CALON**: Calibration mode enabled
0: Normal mode
1: Calibration mode (all switches opened by HW)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **OPAHSM**: Operational amplifier high-speed mode
The operational amplifier must be disable to change this configuration.
0: operational amplifier in normal mode
1: operational amplifier in high-speed mode

Bit 7 Reserved, must be kept at reset value.
Bits 6:5 **VM_SEL**: Inverting input selection
- 00: INM0 connected to OPAMP INM input
- 01: INM1 connected to OPAMP NM input
- 10: Feedback resistor is connected to OPAMP INM input (PGA mode), Inverting input selection is depends on the PGA_GAIN setting
- 11: opamp_out connected to OPAMP INM input (Follower mode)

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **VP_SEL**: Non inverted input selection
- 00: GPIO connected to OPAMPx_VINP
- 01: dac_outx connected to OPAMPx_VINP
- 10: Reserved
- 11: Reserved

Bit 1 **FORCE_VP**: Force internal reference on VP (reserved for test)
- 0: Normal operating mode. Non-inverting input connected to inputs.
- 1: Calibration verification mode: Non-inverting input connected to calibration reference voltage.

Bit 0 **OPAEN**: Operational amplifier Enable
- 0: operational amplifier disabled
- 1: operational amplifier enabled

*Note:* If OPAMP1 is unconnected in a specific package, it must remain disabled (keep OPAMP1_CSR register default value).

### 32.6.2 OPAMP1 trimming register in normal mode (OPAMP1_OTR)

Address: 0x04

Reset value: 0x0000 XXXX (factory trimmed values)

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

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

| rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **TRIMOFFSETP[4:0]**: Trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMOFFSETN[4:0]**: Trim for NMOS differential pairs
### 32.6.3 OPAMP1 trimming register in high-speed mode (OPAMP1_HSOTR)

Address: 0x08
Reset value: 0x0000 XXXX (factory trimmed values)

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

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **TRIMHSOFFSETP[4:0]**: High-speed mode trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMHSOFFSETN[4:0]**: High-speed mode trim for NMOS differential pairs

### 32.6.4 OPAMP option register (OPAMP_OR)

Address: 0x0C
Reset value: 0x0000 0000

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

Bits 31:0 Reserved, must be kept at reset value.

### 32.6.5 OPAMP2 control/status register (OPAMP2_CSR)

Address: 0x10
Reset value: 0x0000 0000

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

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

Bits 21:12 **PGA_GAIN**: PGA gain

Bits 11:0 **CALSEL**: Calibration select

Bits 9:0 **CALON**: Calibration enable

Bits 8:0 **OPAHSM**: Output pins high side enable

Bits 7:0 **VM_SEL**: Voltage mode select

Bits 5:0 **VP_SEL**: Voltage pins select

Bits 3:0 **FORCEVP**: Force voltage pins

Bits 1:0 **OPAEN**: Output pins enable
Bit 31 Reserved, must be kept at reset value.

Bit 30 **CALOUT**: Operational amplifier calibration output
OPAMP output status flag. During the calibration mode, OPAMP is used as comparator.
0: Non-inverting < inverting
1: Non-inverting > inverting

Bit 29 **TSTREF**: OPAMP calibration reference voltage output control (reserved for test)
0: INTVREF of OPAMP is not output
1: INTVREF of OPAMP is output

Bits 28:19 Reserved, must be kept at reset value.

Bit 18 **USERTRIM**: User trimming enable
This bit allows to switch from ‘factory’ AOP offset trimmed values to ‘user’ AOP offset trimmed values.
This bit is active for both mode normal and high-power.
0: ‘factory’ trim code used
1: ‘user’ trim code used

Bits 17:14 **PGA_GAIN**: Operational amplifier Programmable amplifier gain value
0000: Non-inverting internal Gain 2, VREF- referenced
0001: Non-inverting internal Gain 4, VREF- referenced
0010: Non-inverting internal Gain 8, VREF- referenced
0011: Non-inverting internal Gain 16, VREF- referenced
0100: Non-inverting internal Gain 2 with filtering on INM0, VREF- referenced
0101: Non-inverting internal Gain 4 with filtering on INM0, VREF- referenced
0110: Non-inverting internal Gain 8 with filtering on INM0, VREF- referenced
0111: Non-inverting internal Gain 16 with filtering on INM0, VREF- referenced
1000: Inverting gain=-1/ Non-inverting gain =2 with INM0 node for input or bias
1001: Inverting gain=-3/ Non-inverting gain =4 with INM0 node for input or bias
1010: Inverting gain=-7/ Non-inverting gain =8 with INM0 node for input or bias
1011: Inverting gain=-15/ Non-inverting gain =16 with INM0 node for input or bias
1100: Inverting gain=-1/ Non-inverting gain =2 with INM0 node for input or bias, INM1 node for filtering
1101: Inverting gain=-3/ Non-inverting gain =4 with INM0 node for input or bias, INM1 node for filtering
1110: Inverting gain=-7/ Non-inverting gain =8 with INM0 node for input or bias, INM1 node for filtering
1111: Inverting gain=-15/ Non-inverting gain =16 with INM0 node for input or bias, INM1 node for filtering

Bits 13:12 **CALSEL**: Calibration selection
It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP= 1.
00: 0.033*VDDA applied on OPAMP inputs
01: 0.1*VDDA applied on OPAMP inputs (for PMOS calibration)
10: 0.5*VDDA applied on OPAMP inputs
11: 0.9*VDDA applied on OPAMP inputs (for NMOS calibration)

Bit 11 **CALON**: Calibration mode enabled
0: Normal mode
1: Calibration mode (all switches opened by HW)

Bits 10:9 Reserved, must be kept at reset value.
32.6.6 OPAMP2 trimming register in normal mode (OPAMP2_OTR)

Bit 8 OPAHSM: Operational amplifier high-speed mode
The operational amplifier must be disable to change this configuration.
0: operational amplifier in normal mode
1: operational amplifier in high-speed mode

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 VM_SEL: Inverting input selection
00: INM0 connected to OPAMP INM input
01: INM1 connected to OPAMP INM input
10: Feedback resistor is connected to OPAMP INM input (PGA mode), Inverting input selection is depends on the PGA_GAIN setting
11: opamp_out connected to OPAMP INM input (Follower mode)

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 VP_SEL: Non inverted input selection
00: GPIO connected to OPAMPx_VINP
01: DAC connected to OPAMPx_VINP
10: Reserved
11: Reserved

Bit 1 FORCE_VP: Force internal reference on VP (reserved for test)
0: Normal operating mode. Non-inverting input connected to inputs.
1: Calibration verification mode: Non-inverting input connected to calibration reference voltage.

Bit 0 OPAEN: Operational amplifier Enable
0: operational amplifier disabled
1: operational amplifier enabled

Note: If OPAMP2 is unconnected in a specific package, it must remain disabled (keep OPAMP2_CSR register default value).

32.6.6 OPAMP2 trimming register in normal mode (OPAMP2_OTR)

Address: 0x14

Reset value: 0x0000 XXXX (factory trimmed values)
32.6.7  **OPAMP2 trimming register in high-speed mode (OPAMP2_HSOTR)**

Address: 0x18  
Reset value: 0x0000 XXXX (factory trimmed values)

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

Bits 31:13  Reserved, must be kept at reset value.

Bits 12:8  **TRIMHSOFFSETP[4:0]**: High-speed mode trim for PMOS differential pairs

Bits 7:5  Reserved, must be kept at reset value.

Bits 4:0  **TRIMHSOFFSETN[4:0]**: High-speed mode trim for NMOS differential pairs
### 32.6.8 OPAMP register map

#### Table 235. OPAMP register map and reset values

| Offset | Register name  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | OPAMP1_CSR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | OPAMP1_OTR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | OPAMP1_HSOTR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | OPAMP2_CSR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | OPAMP2_OTR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x14   | OPAMP2_HSOTR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

1. Factory trimmed values.

Refer to [Section 2.3 on page 128](#) for the register boundary addresses.
33 Digital filter for sigma delta modulators (DFSDM)

33.1 Introduction

Digital filter for sigma delta modulators (DFSDM) is a high-performance module dedicated to interface external Σ∆ modulators. It is featuring up to 8 external digital serial interfaces (channels) and up to 8 digital filters with flexible Sigma Delta stream digital processing options to offer up to 24-bit final ADC resolution. DFSDM also features optional parallel data stream input from internal ADC peripherals or from device memory.

An external Σ∆ modulator provides digital data stream of converted analog values from the external Σ∆ modulator analog input. This digital data stream is sent into a DFSDM input channel through a serial interface. DFSDM supports several standards to connect various Σ∆ modulator outputs: SPI interface and Manchester coded 1-wire interface (both with adjustable parameters). DFSDM module supports the connection of up to 8 multiplexed input digital serial channels which are shared with up to 8 DFSDM modules. DFSDM module also supports alternative parallel data inputs from up to 8 internal 16-bit data channels (from internal ADCs or from device memory).

DFSDM is converting an input data stream into a final digital data word which represents an analog input value on a Σ∆ modulator analog input. The conversion is based on a configurable digital process: the digital filtering and decimation of the input serial data stream.

The conversion speed and resolution are adjustable according to configurable parameters for digital processing: filter type, filter order, length of filter, integrator length. The maximum output data resolution is up to 24 bits. There are two conversion modes: single conversion mode and continuous mode. The data can be automatically stored in a system RAM buffer through DMA, thus reducing the software overhead.

A flexible timer triggering system can be used to control the start of conversion of DFSDM. This timing control is capable of triggering simultaneous conversions or inserting a programmable delay between conversions.

DFSDM features an analog watchdog function. Analog watchdog can be assigned to any of the input channel data stream or to final output data. Analog watchdog has its own digital filtering of input data stream to reach the required speed and resolution of watched data.

To detect short-circuit in control applications, there is a short-circuit detector. This block watches each input channel data stream for occurrence of stable data for a defined time duration (several 0’s or 1’s in an input data stream).

An extremes detector block watches final output data and stores maximum and minimum values from the output data values. The extremes values stored can be restarted by software.

Two power modes are supported: normal mode and stop mode.
33.2 **DFSDM main features**

- Up to 8 multiplexed input digital serial channels:
  - configurable SPI interface to connect various ΣΔ modulators
  - configurable Manchester coded 1 wire interface support
  - clock output for ΣΔ modulator(s)
- Alternative inputs from up to 8 internal digital parallel channels:
  - inputs with up to 16 bit resolution
  - internal sources: ADCs data or memory (CPU/DMA write) data streams
- Adjustable digital signal processing:
  - Sincx filter: filter order/type (1..5), oversampling ratio (up to 1..1024)
  - integrator: oversampling ratio (1..256)
- Up to 24-bit output data resolution:
  - right bit-shifter on final data (0..31 bits)
- Signed output data format
- Automatic data offset correction (offset stored in register by user)
- Continuous or single conversion
- Start-of-conversion synchronization with:
  - software trigger
  - internal timers
  - external events
  - start-of-conversion synchronously with first DFSDM filter (DFSDM_FLT0)
- Analog watchdog feature:
  - low value and high value data threshold registers
  - own configurable Sinc³ digital filter (order = 1..3, oversampling ratio = 1..32)
  - input from output data register or from one or more input digital serial channels
  - continuous monitoring independently from standard conversion
- Short-circuit detector to detect saturated analog input values (bottom and top ranges):
  - up to 8-bit counter to detect 1..256 consecutive 0's or 1's on input data stream
  - monitoring continuously each channel (8 serial channel transceiver outputs)
- Break generation on analog watchdog event or short-circuit detector event
- Extremes detector:
  - store minimum and maximum values of output data values
  - refreshed by software
- Pulse skipper feature to support beamforming applications (delay line like behavior)
- DMA may be used to read the conversion data
- Interrupts: end of conversion, overrun, analog watchdog, short-circuit, channel clock absence
- “regular” or “injected” conversions:
  - “regular” conversions can be requested at any time or even in continuous mode without having any impact on the timing of “injected” conversions
  - “injected” conversions for precise timing and with high conversion priority
33.3 DFSDM implementation

This section describes the configuration implemented in DFSDMx.

Table 236. DFSDMx implementation

<table>
<thead>
<tr>
<th>DFSDM features</th>
<th>DFSDM1</th>
<th>DFSDM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Number of filters</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Input from internal ADC</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Supported trigger sources</td>
<td>$32^{(1)}$</td>
<td>$32^{(1)}$</td>
</tr>
<tr>
<td>Pulses skipper</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>ID registers support</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. Refer to Table 241: DFSDM triggers connection for available trigger sources.
33.4 DFSDM functional description

33.4.1 DFSDM block diagram

Figure 249. Single DFSDM block diagram
1. This example shows 8 DFSDM filters and 8 input channels (max. configuration).
2. Not available for DFSDM2. See Table 236: DFSDMx implementation for available features in DFSDM2.

### 33.4.2 DFSDM pins and internal signals

#### Table 237. DFSDM1 external pins

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal Type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>Power supply</td>
<td>Digital power supply.</td>
</tr>
<tr>
<td>VSS</td>
<td>Power supply</td>
<td>Digital ground power supply.</td>
</tr>
<tr>
<td>CKIN[7:0]</td>
<td>Clock input</td>
<td>Clock signal provided from external ΣΔ modulator. FT input.</td>
</tr>
<tr>
<td>DATIN[7:0]</td>
<td>Data input</td>
<td>Data signal provided from external ΣΔ modulator. FT input.</td>
</tr>
<tr>
<td>CKOUT</td>
<td>Clock output</td>
<td>Clock output to provide clock signal into external ΣΔ modulator.</td>
</tr>
<tr>
<td>EXTRG[1:0]</td>
<td>External trigger signal</td>
<td>Input trigger from two EXTI signals to start analog conversion (from GPIOs: EXTI11, EXTI15).</td>
</tr>
</tbody>
</table>

#### Table 238. DFSDM2 external pins

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal Type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>Power supply</td>
<td>Digital power supply.</td>
</tr>
<tr>
<td>VSS</td>
<td>Power supply</td>
<td>Digital ground power supply.</td>
</tr>
<tr>
<td>CKIN[1:0]</td>
<td>Clock input</td>
<td>Clock signal provided from external ΣΔ modulator. FT input.</td>
</tr>
<tr>
<td>DATIN[1:0]</td>
<td>Data input</td>
<td>Data signal provided from external ΣΔ modulator. FT input.</td>
</tr>
<tr>
<td>CKOUT</td>
<td>Clock output</td>
<td>Clock output to provide clock signal into external ΣΔ modulator.</td>
</tr>
<tr>
<td>EXTRG[1:0]</td>
<td>External trigger signal</td>
<td>Input trigger from two EXTI signals to start analog conversion (from GPIOs: EXTI11, EXTI15).</td>
</tr>
</tbody>
</table>

#### Table 239. DFSDM1 internal signals

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal Type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfsdm_jtrg[31:0]</td>
<td>Internal/ external trigger signal</td>
<td>Input trigger from internal/external trigger sources in order to start analog conversion (from internal sources: synchronous input, from external sources: asynchronous input with synchronization). See Table 242 for details.</td>
</tr>
<tr>
<td>dfsdm_break[3:0]</td>
<td>break signal output</td>
<td>Break signals event generation from Analog watchdog or short-circuit detector</td>
</tr>
<tr>
<td>dfsdm_dma[7:0]</td>
<td>DMA request signal</td>
<td>DMA request signal from each DFSDM_FLTx (x=0..7): end of injected conversion event.</td>
</tr>
<tr>
<td>dfsdm_it[7:0]</td>
<td>Interrupt request signal</td>
<td>Interrupt signal for each DFSDM_FLTx (x=0..7)</td>
</tr>
<tr>
<td>dfsdm_dat_adc[15:0]</td>
<td>ADC input data</td>
<td>Up to 4 internal ADC data buses as parallel inputs.</td>
</tr>
</tbody>
</table>
### Table 240. DFSDM2 internal signals

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal Type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfsdm_jtrg[31:0]</td>
<td>Internal/external trigger signal</td>
<td>Input trigger from internal/external trigger sources in order to start analog conversion (from internal sources: synchronous input, from external sources: asynchronous input with synchronization). See Table 242 for details.</td>
</tr>
<tr>
<td>dfsdm_break[3:0]</td>
<td>break signal output</td>
<td>Break signals event generation from Analog watchdog or short-circuit detector</td>
</tr>
<tr>
<td>dfsdm_dma</td>
<td>DMA request signal</td>
<td>DMA request signal from DFSDM_FLT0: end of injected conversion event.</td>
</tr>
<tr>
<td>dfsdm_it</td>
<td>Interrupt request signal</td>
<td>Interrupt signal for each DFSDM_FLT0</td>
</tr>
</tbody>
</table>

### Table 241. DFSDM triggers connection

<table>
<thead>
<tr>
<th>Trigger name</th>
<th>Trigger source DFSDM1</th>
<th>Trigger source DFSDM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfsdm_jtrg0</td>
<td>TIM1_TRGO</td>
<td>TIM1_TRGO2</td>
</tr>
<tr>
<td>dfsdm_jtrg1</td>
<td>TIM1_TRGO2</td>
<td>TIM8_TRGO</td>
</tr>
<tr>
<td>dfsdm_jtrg2</td>
<td>TIM8_TRGO</td>
<td>TIM8_TRGO2</td>
</tr>
<tr>
<td>dfsdm_jtrg3</td>
<td>TIM3_TRGO</td>
<td>TIM4_TRGO</td>
</tr>
<tr>
<td>dfsdm_jtrg4</td>
<td>TIM3_TRGO2</td>
<td>TIM16_OC1</td>
</tr>
<tr>
<td>dfsdm_jtrg5</td>
<td>TIM4_TRGO</td>
<td>TIM6_TRGO</td>
</tr>
<tr>
<td>dfsdm_jtrg6</td>
<td>TIM6_TRGO</td>
<td>TIM7_TRGO</td>
</tr>
<tr>
<td>dfsdm_jtrg7</td>
<td>TIM7_TRGO</td>
<td>TIM16_OC1</td>
</tr>
<tr>
<td>dfsdm_jtrg8</td>
<td>TIM16_OC1</td>
<td>TIM7_TRGO</td>
</tr>
<tr>
<td>dfsdm_jtrg9</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>dfsdm_jtrg24</td>
<td>EXTI11</td>
<td>EXTI11</td>
</tr>
<tr>
<td>dfsdm_jtrg25</td>
<td>EXTI15</td>
<td>EXTI15</td>
</tr>
<tr>
<td>dfsdm_jtrg26</td>
<td>LPTIMER1</td>
<td>LPTIMER1</td>
</tr>
<tr>
<td>dfsdm_jtrg27</td>
<td>LPTIMER2</td>
<td>LPTIMER2</td>
</tr>
<tr>
<td>dfsdm_jtrg28</td>
<td>LPTIMER3</td>
<td>LPTIMER3</td>
</tr>
<tr>
<td>dfsdm_jtrg29</td>
<td>COMP1_OUT</td>
<td>COMP1_OUT</td>
</tr>
<tr>
<td>dfsdm_jtrg30</td>
<td>COMP2_OUT</td>
<td>COMP2_OUT</td>
</tr>
<tr>
<td>dfsdm_jtrg31</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### Table 242. DFSDM1 break connection

<table>
<thead>
<tr>
<th>Break name</th>
<th>DFSDM1 break destination</th>
<th>DFSDM2 break destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfsdm_break[0]</td>
<td>TIM1/TIM15 break</td>
<td>LPTIM3_ETR</td>
</tr>
<tr>
<td>dfsdm_break[1]</td>
<td>TIM1 break2/TIM16 break</td>
<td>-</td>
</tr>
</tbody>
</table>
### 33.4.3 DFSDM reset and clocks

#### DFSDM on-off control

The DFSDM interface is globally enabled by setting DFSDMEN=1 in the DFSDM_CH0CFGR1 register. Once DFSDM is globally enabled, all input channels (y=0..7) and digital filters DFSDM_FLTx (x=0..7) start to work if their enable bits are set (channel enable bit CHEN in DFSDM_CHyCFGR1 and DFSDM_FLTx enable bit DFEN in DFSDM_FLTxCR1).

Digital filter x DFSDM_FLTx (x=0..7) is enabled by setting DFEN=1 in the DFSDM_FLTxCR1 register. Once DFSDM_FLTx is enabled (DFEN=1), both Sincx digital filter unit and integrator unit are reinitialized.

By clearing DFEN, any conversion which may be in progress is immediately stopped and DFSDM_FLTx is put into stop mode. All register settings remain unchanged except DFSDM_FLTxAWSR and DFSDM_FLTxISR (which are reset).

Channel y (y=0..7) is enabled by setting CHEN=1 in the DFSDM_CHyCFGR1 register. Once the channel is enabled, it receives serial data from the external Σ∆ modulator or parallel internal data sources (ADCs or CPU/DMA wire from memory).

DFSDM must be globally disabled (by DFSDMEN=0 in DFSDM_CH0CFGR1) before stopping the system clock to enter in the STOP mode of the device.

#### DFSDM clocks

The internal DFSDM clock f_{DFSDMCLK}, which is used to drive the channel transceivers, digital processing blocks (digital filter, integrator) and next additional blocks (analog watchdog, short-circuit detector, extremes detector, control block) is generated by the RCC block and is derived from the system clock SYSCLK or peripheral clock PCLK2 (see Section 8.7.18: RCC CPU domain kernel clock configuration register (RCC_CDCCIP1R)).

The DFSDM clock is automatically stopped in stop mode (if DFEN = 0 for all DFSDM_FLTx, x=0..7).

The DFSDM serial channel transceivers can receive an external serial clock to sample an external serial data stream. The internal DFSDM clock must be at least 4 times faster than the external serial clock if standard SPI coding is used, and 6 times faster than the external serial clock if Manchester coding is used.

DFSDM can provide one external output clock signal to drive external Σ∆ modulator(s) clock input(s). It is provided on CKOUT pin. This output clock signal must be in the range specified in given device datasheet and is derived from DFSDM clock or from audio clock (see CKOUTSRC bit in DFSDM_CH0CFGR1 register) by programmable divider in the range 2 - 256 (CKOUTDIV in DFSDM_CH0CFGR1 register). Audio clock source is SAI1 clock selected by SAI1SEL[1:0] field in RCC configuration (see ).
33.4.4 Serial channel transceivers

There are 8 multiplexed serial data channels which can be selected for conversion by each filter or Analog watchdog or Short-circuit detector. Those serial transceivers receive data stream from external \(\Sigma\Delta\) modulator. Data stream can be sent in SPI format or Manchester coded format (see SITP[1:0] bits in DFSDM_CHyCFGR1 register).

The channel is enabled for operation by setting CHEN=1 in DFSDM_CHyCFGR1 register.

Channel inputs selection

Serial inputs (data and clock signals) from DATINy and CKINy pins can be redirected from the following channel pins. This serial input channel redirection is set by CHINSEL bit in DFSDM_CHyCFGR1 register.

Channel redirection can be used to collect audio data from PDM (pulse density modulation) stereo microphone type. PDM stereo microphone has one data and one clock signal. Data signal provides information for both left and right audio channel (rising clock edge samples for left channel and falling clock edge samples for right channel).

Configuration of serial channels for PDM microphone input:

- PDM microphone signals (data, clock) will be connected to DFSDM input serial channel \(y\) (DATINy, CKOUT) pins.
- Channel \(y\) will be configured: CHINSEL = 0 (input from given channel pins: DATINy, CKINy).
- Channel \((y-1)\) (modulo 8) will be configured: CHINSEL = 1 (input from the following channel \((y-1)+1\) pins: DATINy, CKINy).
- Channel \(y\): SITP[1:0] = 0 (rising edge to strobe data) => left audio channel on channel \(y\).
- Channel \((y-1)\): SITP[1:0] = 1 (falling edge to strobe data) => right audio channel on channel \(y-1\).
- Two DFSDM filters will be assigned to channel \(y\) and channel \((y-1)\) (to filter left and right channels from PDM microphone).
Output clock generation

A clock signal can be provided on CKOUT pin to drive external ΣΔ modulator clock inputs. The frequency of this CKOUT signal is derived from DFSDM clock or from audio clock (see CKOUTSRC bit in DFSDM_CH0CFGR1 register) divided by a predivider (see CKOUTDIV bits in DFSDM_CH0CFGR1 register). If the output clock is stopped, then CKOUT signal is set to low state (output clock can be stopped by CKOUTDIV=0 in DFSDM_CHyCFGR1 register or by DFSDMEN=0 in DFSDM_CH0CFGR1 register). The output clock stopping is performed:

- 4 system clocks after DFSDMEN is cleared (if CKOUTSRC=0)
- 1 system clock and 3 audio clocks after DFSDMEN is cleared (if CKOUTSRC=1)

Before changing CKOUTSRC the software has to wait for CKOUT being stopped to avoid glitch on CKOUT pin. The output clock signal frequency must be in the range 0 - 20 MHz.
SPI data input format operation

In SPI format, the data stream is sent in serial format through data and clock signals. Data signal is always provided from DATINy pin. A clock signal can be provided externally from CKINy pin or internally from a signal derived from the CKOUT signal source.

In case of external clock source selection (SPICKSEL[1:0]=0) data signal (on DATINy pin) is sampled on rising or falling clock edge (of CKINy pin) according SITP[1:0] bits setting (in DFSDM_CYCFGR1 register).

Internal clock sources - see SPICKSEL[1:0] in DFSDM_CYCFGR1 register:

- CKOUT signal:
  - For connection to external ΣΔ modulator which uses directly its clock input (from CKOUT) to generate its output serial communication clock.
  - Sampling point: on rising/falling edge according SITP[1:0] setting.

- CKOUT/2 signal (generated on CKOUT rising edge):
  - For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input rising edge).
  - Sampling point: on each second CKOUT falling edge.

- CKOUT/2 signal (generated on CKOUT falling edge):
  - For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input falling edge).
  - Sampling point: on each second CKOUT rising edge.

Note: An internal clock source can only be used when the external ΣΔ modulator uses CKOUT signal as a clock input (to have synchronous clock and data operation).

Internal clock source usage can save CKINy pin connection (CKINy pins can be used for other purpose).

The clock source signal frequency must be in the range 0 - 20 MHz for SPI coding and less than f_{DFSDMCLK}/4.

Manchester coded data input format operation

In Manchester coded format, the data stream is sent in serial format through DATINy pin only. Decoded data and clock signal are recovered from serial stream after Manchester decoding. There are two possible settings of Manchester codings (see SITP[1:0] bits in DFSDM_CYCFGR1 register):

- signal rising edge = log 0; signal falling edge = log 1
- signal rising edge = log 1; signal falling edge = log 0

The recovered clock signal frequency for Manchester coding must be in the range 0 - 10 MHz and less than f_{DFSDMCLK}/6.

To correctly receive Manchester coded data, the CKOUTDIV divider (in DFSDM_CY0CFGR1 register) must be set with respect to expected Manchester data rate according formula:

\[ ((\text{CKOUTDIV} + 1) \times T_{\text{SYSCLK}}) < T_{\text{Manchester clock}} < (2 \times \text{CKOUTDIV} \times T_{\text{SYSCLK}}) \]
Figure 251. Channel transceiver timing diagrams

- SPI timing: SPIKSEL = 0
  - SITP = 00
  - SITP = 01

- SPI timing: SPIKSEL = 1, 2, 3
  - SPICKSEL = 1
  - SPICKSEL = 2
  - SPICKSEL = 3

- Manchester timing
  - SITP = 0
  - SITP = 1
  - SITP = 2
  - SITP = 3

Recovered clock and recovered data
Clock absence detection

Channels serial clock inputs can be checked for clock absence/presence to ensure the correct operation of conversion and error reporting. Clock absence detection can be enabled or disabled on each input channel \( y \) by bit CKABEN in DFSDM_CHyCFGR1 register. If enabled, then this clock absence detection is performed continuously on a given channel. A clock absence flag is set (CKABF\( [y] = 1 \)) and an interrupt can be invoked (if CKABIE=1) in case of an input clock error (see CKABF\([7:0]\) in DFSDM_FLT0ISR register and CKABEN in DFSDM_CHyCFGR1). After a clock absence flag clearing (by CLRCKABF in DFSDM_FLT0ICR register), the clock absence flag is refreshed. Clock absence status bit CKABF\([y]\) is set also by hardware when corresponding channel \( y \) is disabled (if CHEN\([y]\) = 0 then CKABF\([y]\) is held in set state).

When a clock absence event has occurred, the data conversion (and/or analog watchdog and short-circuit detector) provides incorrect data. The user should manage this event and discard given data while a clock absence is reported.

The clock absence feature is available only when the system clock is used for the CKOUT signal (CKOUTSRC=0 in DFSDM_CH0CFGR1 register).

When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF\([y]\) bit (in DFSDM_FLT0ICR register). The software sequence concerning clock absence detection feature should be:

- Enable given channel by CHEN = 1
- Try to clear the clock absence flag (by CLRCKABF = 1) until the clock absence flag is really cleared (CKABF = 0). At this time, the transceiver is synchronized (signal clock is valid) and is able to receive data.
- Enable the clock absence feature CKABEN = 1 and the associated interrupt CKABIE = 1 to detect if the SPI clock is lost or Manchester data edges are missing.

If SPI data format is used, then the clock absence detection is based on the comparison of an external input clock with an output clock generation (CKOUT signal). The external input clock signal into the input channel must be changed at least once per 8 signal periods of CKOUT signal (which is controlled by CKOUTDIV field in DFSDM_CH0CFGR1 register).

**Figure 252. Clock absence timing diagram for SPI**

If Manchester data format is used, then the clock absence means that the clock recovery is unable to perform from Manchester coded signal. For a correct clock recovery, it is first necessary to receive data with 1 to 0 or 0 to 1 transition (see Figure 254 for Manchester synchronization).
The detection of a clock absence in Manchester coding (after a first successful synchronization) is based on changes comparison of coded serial data input signal with output clock generation (CKOUT signal). There must be a voltage level change on DATINy pin during 2 periods of CKOUT signal (which is controlled by CKOUTDIV bits in DFSDM_CH0CFGR1 register). This condition also defines the minimum data rate to be able to correctly recover the Manchester coded data and clock signals.

The maximum data rate of Manchester coded data must be less than the CKOUT signal.

So to correctly receive Manchester coded data, the CKOUTDIV divider must be set according the formula:

\[( (\text{CKOUTDIV} + 1) \times T_{\text{SYSCLK}} < T_{\text{Manchester clock}} < (2 \times \text{CKOUTDIV} \times T_{\text{SYSCLK}}) \]

A clock absence flag is set (CKABF\[y\] = 1) and an interrupt can be invoked (if CKABIE=1) in case of an input clock recovery error (see CKABF\[7:0\] in DFSDM_FLT0ISR register and CKABEN in DFSDM_CHyCFGR1). After a clock absence flag clearing (by CLRCKABF in DFSDM_FLT0ICR register), the clock absence flag is refreshed.

Figure 253. Clock absence timing diagram for Manchester coding
Manchester/SPI code synchronization

The Manchester coded stream must be synchronized the first time after enabling the channel (CHEN=1 in DFSDM_CHyCFGR1 register). The synchronization ends when a data transition from 0 to 1 or from 1 to 0 (to be able to detect valid data edge) is received. The end of the synchronization can be checked by polling CKABF[y]=0 for a given channel after it has been cleared by CLRCKABF[y] in DFSDM_FLT0ICR, following the software sequence detailed hereafter:

CKABF[y] flag is cleared by setting CLRCKABF[y] bit. If channel y is not yet synchronized the hardware immediately set the CKABF[y] flag. Software is then reading back the CKABF[y] flag and if it is set then perform again clearing of this flag by setting CLRCKABF[y] bit. This software sequence (polling of CKABF[y] flag) continues until CKABF[y] flag is set (signalizing that Manchester stream is synchronized). To be able to synchronize/receive Manchester coded data the CKOUTDIV divider (in DFSDM_CH0CFGR1 register) must be set with respect to expected Manchester data rate according the formula below.

\[(\text{CKOUTDIV} + 1) \times T_{\text{SYSCLK}} < T_{\text{Manchester clock}} < 2 \times \text{CKOUTDIV} \times T_{\text{SYSCLK}}\]

SPI coded stream is synchronized after first detection of clock input signal (valid rising/falling edge).

Note: When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y] bit (in DFSDM_FLT0ICR register).
External serial clock frequency measurement

The measuring of a channel serial clock input frequency provides a real data rate from an external Σ∆ modulator, which is important for application purposes.

An external serial clock input frequency can be measured by a timer counting DFSDM clocks (\(f_{DFSDMCLK}\)) during one conversion duration. The counting starts at the first input data clock after a conversion trigger (regular or injected) and finishes by last input data clock before conversion ends (end of conversion flag is set). Each conversion duration (time between first serial sample and last serial sample) is updated in counter CNVCNT[27:0] in register DFSDM_FLTxCNVTIMR when the conversion finishes (JEOCF=1 or REOCF=1). The user can then compute the data rate according to the digital filter settings (FORD, FOSR, IOSR, FAST). The external serial frequency measurement is stopped only if the filter is bypassed (FOSR=0, only integrator is active, CNVCNT[27:0]=0 in DFSDM_FLTxCNVTIMR register).

In case of parallel data input (Section 33.4.6: Parallel data inputs) the measured frequency is the average input data rate during one conversion.
Note: When conversion is interrupted (e.g. by disabling/enabling the selected channel) the interruption time is also counted in CNVCNT[27:0]. Therefore it is recommended to not interrupt the conversion for correct conversion duration result.

Conversion times:

**injected conversion or regular conversion with FAST = 0 (or first conversion if FAST=1):**

for Sinc^x filters (x=1..5):
\[
t = \frac{\text{CNVCNT}}{f_{\text{DFSDMCLK}}} = \frac{[F_{\text{OSR}} * (I_{\text{OSR}} - 1 + F_{\text{ORD}})]}{f_{\text{CKIN}}}
\]

for FastSinc filter:
\[
t = \frac{\text{CNVCNT}}{f_{\text{DFSDMCLK}}} = \frac{[F_{\text{OSR}} * (I_{\text{OSR}} - 1 + 4)]}{f_{\text{CKIN}}}
\]

**regular conversion with FAST = 1 (except first conversion):**

for Sinc and FastSinc filters:
\[
t = \frac{\text{CNVCNT}}{f_{\text{DFSDMCLK}}} = \frac{[F_{\text{OSR}} * I_{\text{OSR}}]}{f_{\text{CKIN}}}
\]

in case if F_{OSR} = FOSR[9:0]+1 = 1 (filter bypassed, active only integrator):
\[
t = \frac{I_{\text{OSR}}}{f_{\text{CKIN}}} \text{ (... but CNVCNT=0)}
\]

where:
- \(f_{\text{CKIN}}\) is the channel input clock frequency (on given channel CKINy pin) or input data rate (in case of parallel data input)
- \(F_{\text{OSR}}\) is the filter oversampling ratio: \(F_{\text{OSR}} = FOSR[9:0]+1\) (see DFSDM_FLTxFCR register)
- \(I_{\text{OSR}}\) is the integrator oversampling ratio: \(I_{\text{OSR}} = IOSR[7:0]+1\) (see DFSDM_FLTxFCR register)
- \(F_{\text{ORD}}\) is the filter order: \(F_{\text{ORD}} = FORD[2:0]\) (see DFSDM_FLTxFCR register)

**Channel offset setting**

Each channel has its own offset setting (in register) which is finally subtracted from each conversion result (injected or regular) from a given channel. Offset correction is performed after the data right bit shift. The offset is stored as a 24-bit signed value in OFFSET[23:0] field in DFSDM_CHyCFGR2 register.

**Data right bit shift**

To have the result aligned to a 24-bit value, each channel defines a number of right bit shifts which will be applied on each conversion result (injected or regular) from a given channel. The data bit shift number is stored in DTRBS[4:0] bits in DFSDM_CHyCFGR2 register.

The right bit-shift is rounding the result to nearest integer value. The sign of shifted result is maintained, in order to have valid 24-bit signed format of result data.

**Pulses skipper**

Purpose of the pulses skipper is to implement delay line like behavior for given input channel(s). Given number of samples from input serial data stream (serial stream only) can be discarded before they enter into the filter. This data discarding is performed by skipping given number of sampling input clock pulses (given serial data samples are then not sampled by filter). The sampling clock is gated by pulses skipper function for given number of clock pulses. When given clock pulses are skipped then the filtering continues for following input data. With comparison to non skipped data stream this operation causes that
the final output sample (and next samples) from filter will be calculated from later input data. This final sample then looks a bit in forward - because it is calculated from newer input samples than the “non-skipped” sample. The final “skipped sample” is converted later because the skipped input data samples must be replaced by followed input data samples. The final data buffers behavior (skipped and non-skipped output data buffers comparison) looks like the non-skipped data stream is a bit delayed - both data buffers will be phase shifted.

Number of clock pulses to be skipped should be written into PLSSKP[5:0] field in DFSDM_CHyDLYR register. Once PLSSKP[5:0] field is written the execution of pulses skipping is started on given channel. PLSSKP[5:0] field can be read in order to check the progress of pulses skipper. When PLSSKP[5:0]=0 means that pulses skipping has been executed.

Up to 63 clock pulses can be skip with a single write operation into PLSSKP[5:0]. If more pulses need to be skipped, then user has to write several times into the PLSSKP[5:0] field. The application software should handle cumulative skipped clock number per each filter.

### 33.4.5 Configuring the input serial interface

The following parameters must be configured for the input serial interface:

- **Output clock predivider.** There is a programmable predivider to generate the output clock from DFSDM clock (2 - 256). It is defined by CKOUTDIV[7:0] bits in DFSDM_CH0CFGR1 register.
- **Serial interface type and input clock phase.** Selection of SPI or Manchester coding and sampling edge of input clock. It is defined by SITP[1:0] bits in DFSDM_CHyCFGR1 register.
- **Input clock source.** External source from CKINy pin or internal from CKOUT pin. It is defined by SPICKSEL[1:0] field in DFSDM_CHyCFGR1 register.
- **Final data right bit-shift.** Defines the final data right bit shift to have the result aligned to a 24-bit value. It is defined by DTRBS[4:0] in DFSDM_CHyCFGR2 register.
- **Channel offset per channel.** Defines the analog offset of a given serial channel (offset of connected external ΣΔ modulator). It is defined by OFFSET[23:0] bits in DFSDM_CHyCFGR2 register.
- **short-circuit detector and clock absence per channel enable.** To enable or disable the short-circuit detector (by SCDEN bit) and the clock absence monitoring (by CKABEN bit) on a given serial channel in register DFSDM_CHyCFGR1.
- **Analog watchdog filter and short-circuit detector threshold settings.** To configure channel analog watchdog filter parameters and channel short-circuit detector parameters. Configurations are defined in DFSDM_CHyAWSCDR register.

### 33.4.6 Parallel data inputs

Each input channel provides a register for 16-bit parallel data input (besides serial data input). Each 16-bit parallel input can be sourced from internal data sources only:

- internal ADC results
- direct CPU/DMA writing.

The selection for using serial or parallel data input for a given channel is done by field DATMPX[1:0] of DFSDM_CHyCFGR1 register. In DATMPX[1:0] is also defined the parallel data source: internal ADC or direct write by CPU/DMA.
Each channel contains a 32-bit data input register DFSDM_CHyDATINR in which it can be written a 16-bit data. Data are in 16-bit signed format. Those data can be used as input to the digital filter which is accepting 16-bit parallel data.

If serial data input is selected (DATMPX[1:0] = 0), the DFSDM_CHyDATINR register is write protected.

**Input from internal ADC**

In case of ADC data parallel input (DATMPX[1:0]=1) the ADC[y+1] result is assigned to channel y input (ADC1 is filling DFSDM_CHDATIN0R register, ADC2 is filling DFSDM_CHDATIN1R register, ... , ADC8 is filling DFSDM_CHDATIN7R register). End of conversion event from ADC[y+1] causes update of channel y data (parallel data from ADC[y+1] are put as next sample to digital filter). Data from ADC[y+1] is written into DFSDM_CHyDATINR register (field INDAT0[15:0]) when end of conversion event occurred.

The setting of data packing mode (DATPACK[1:0] in the DFSDM_CHyCFGR1 register) has no effect in case of ADC data input.

*Note: Extension of ADC specification: in case the internal ADC is configured in interleaved mode (e.g. ADC1 together with ADC2 - see ADC specification) then each result from ADC1 or from ADC2 will come to the same 16-bit bus - to the bus of ADC1 - which is coming into DFSDM channel 0 (fixed connection). So there will be double input data rate into DFSDM channel 0 (even samples come from ADC1 and odd samples from ADC2). Channel 1 associated with ADC2 will be free.*

**Input from memory (direct CPU/DMA write)**

The direct data write into DFSDM_CHyDATINR register by CPU or DMA (DATMPX[1:0]=2) can be used as data input in order to process digital data streams from memory or peripherals.

Data can be written by CPU or DMA into DFSDM_CHyDATINR register:

1. **CPU data write:**
   Input data are written directly by CPU into DFSDM_CHyDATINR register.

2. **DMA data write:**
   The DMA should be configured in memory-to-memory transfer mode to transfer data from memory buffer into DFSDM_CHyDATINR register. The destination memory address is the address of DFSDM_CHyDATINR register. Data are transferred at DMA transfer speed from memory to DFSDM parallel input.
   
   This DMA transfer is different from DMA used to read DFSDM conversion results. Both DMA can be used at the same time - first DMA (configured as memory-to-memory transfer) for input data writings and second DMA (configured as peripheral-to-memory transfer) for data results reading.

   The accesses to DFSDM_CHyDATINR can be either 16-bit or 32-bit wide, allowing to load respectively one or two samples in one write operation. 32-bit input data register (DFSDM_CHyDATINR) can be filled with one or two 16-bit data samples, depending on the data packing operation mode defined in field DATPACK[1:0] of DFSDM_CHyCFGR1 register:

   1. **Standard mode (DATPACK[1:0]=0):**
      Only one sample is stored in field INDAT0[15:0] of DFSDM_CHyDATINR register which is used as input data for channel y. The upper 16 bits (INDAT1[15:0]) are ignored and write protected. The digital filter must perform one input sampling (from INDAT0[15:0])
to empty data register after it has been filled by CPU/DMA. This mode is used together with 16-bit CPU/DMA access to DFSDM_CHyDATINR register to load one sample per write operation.

2. Interleaved mode (DATPACK[1:0]=1):
DFSDM_CHyDATINR register is used as a two sample buffer. The first sample is stored in INDAT0[15:0] and the second sample is stored in INDAT1[15:0]. The digital filter must perform two input samplings from channel y to empty DFSDM_CHyDATINR register. This mode is used together with 32-bit CPU/DMA access to DFSDM_CHyDATINR register to load two samples per write operation.

3. Dual mode (DATPACK[1:0]=2):
Two samples are written into DFSDM_CHyDATINR register. The data INDAT0[15:0] is for channel y, the data in INDAT1[15:0] is for channel (y+1). The data in INDAT1[15:0] is automatically copied INDAT0[15:0] of the following (y+1) channel data register DFSDM_CH[y+1]DATINR. The digital filters must perform two samplings - one from channel y and one from channel (y+1) - in order to empty DFSDM_CHyDATINR registers.

Dual mode setting (DATPACK[1:0]=2) is available only on even channel numbers (y = 0, 2, 4, 6). If odd channel (y = 1, 3, 5, 7) is set to Dual mode then both INDAT0[15:0] and INDAT1[15:0] parts are write protected for this channel. If even channel is set to Dual mode then the following odd channel must be set into Standard mode (DATPACK[1:0]=0) for correct cooperation with even channels.

See Figure 255 for DFSDM_CHyDATINR registers data modes and assignments of data samples to channels.

![Figure 255. DFSDM_CHyDATINR registers operation modes and assignment](image)

The write into DFSDM_CHyDATINR register to load one or two samples must be performed after the selected input channel (channel y) is enabled for data collection (starting conversion for channel y). Otherwise written data are lost for next processing.

For example: for single conversion and interleaved mode, do not start writing pair of data samples into DFSDM_CHyDATINR before the single conversion is started (any data present in the DFSDM_CHyDATINR before starting a conversion is discarded).
33.4.7 Channel selection

There are 8 multiplexed channels which can be selected for conversion using the injected channel group and/or using the regular channel.

The **injected channel group** is a selection of any or all of the 8 channels. JCHG[7:0] in the DFSDM_FLTxCBCHGR register selects the channels of the injected group, where JCHG[y]=1 means that channel y is selected.

Injected conversions can operate in scan mode (JSCAN=1) or single mode (JSCAN=0). In scan mode, each of the selected channels is converted, one after another. The lowest channel (channel 0, if selected) is converted first, followed immediately by the next higher channel until all the channels selected by JCHG[7:0] have been converted. In single mode (JSCAN=0), only one channel from the selected channels is converted, and the channel selection is moved to the next channel. Writing to JCHG[7:0] if JSCAN=0 resets the channel selection to the lowest selected channel.

Injected conversions can be launched by software or by a trigger. They are never interrupted by regular conversions.

The **regular channel** is a selection of just one of the 8 channels. RCH[2:0] in the DFSDM_FLTxCBCR1 register indicates the selected channel.

Regular conversions can be launched only by software (not by a trigger). A sequence of continuous regular conversions is temporarily interrupted when an injected conversion is requested.

Performing a conversion on a disabled channel (CHEN=0 in DFSDM_CHyCFGR1 register) causes that the conversion will never end - because no input data is provided (with no clock signal). In this case, it is necessary to enable a given channel (CHEN=1 in DFSDM_CHyCFGR1 register) or to stop the conversion by DFEN=0 in DFSDM_FLTxCBCR1 register.

33.4.8 Digital filter configuration

DFSDM contains a Sinc\(x\) type digital filter implementation. This Sinc\(x\) filter performs an input digital data stream filtering, which results in decreasing the output data rate (decimation) and increasing the output data resolution. The Sinc\(x\) digital filter is configurable in order to reach the required output data rates and required output data resolution. The configurable parameters are:

- Filter order/type: (see FORD[2:0] bits in DFSDM_FLTxCBFCR register):
  - FastSinc
  - Sinc\(^1\)
  - Sinc\(^2\)
  - Sinc\(^3\)
  - Sinc\(^4\)
  - Sinc\(^5\)

- Filter oversampling/decimation ratio (see FOSR[9:0] bits in DFSDM_FLTxCBFCR register):
  - FOSR = 1-1024  - for FastSinc filter and Sinc\(x\) filter \(x = F_{ORD} = 1..3\)
  - FOSR = 1-215   - for Sinc\(x\) filter \(x = F_{ORD} = 4\)
  - FOSR = 1-73    - for Sinc\(x\) filter \(x = F_{ORD} = 5\)
The filter has the following transfer function (impulse response in H domain):

- Sinc\(^x\) filter type: \( H(z) = \left( \frac{1 - z^{-\text{FOSR}}} {1 - z^{-1}} \right)^x \)

- FastSinc filter type: \( H(z) = \left( \frac{1 - z^{-\text{FOSR}}} {1 - z^{-1}} \right)^2 \cdot \left( 1 + z^{-2 \cdot \text{FOSR}} \right) \)

Figure 256. Example: Sinc\(^3\) filter response

Table 243. Filter maximum output resolution (peak data values from filter output) for some FOSR values

<table>
<thead>
<tr>
<th>FOSR</th>
<th>Sinc(^1)</th>
<th>Sinc(^2)</th>
<th>FastSinc</th>
<th>Sinc(^3)</th>
<th>Sinc(^4)</th>
<th>Sinc(^5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>+/- x</td>
<td>+/- x(^2)</td>
<td>+/- 2x(^2)</td>
<td>+/- x(^3)</td>
<td>+/- x(^4)</td>
<td>+/- x(^5)</td>
</tr>
<tr>
<td>4</td>
<td>+/- 4</td>
<td>+/- 16</td>
<td>+/- 32</td>
<td>+/- 64</td>
<td>+/- 256</td>
<td>+/- 1024</td>
</tr>
<tr>
<td>8</td>
<td>+/- 8</td>
<td>+/- 64</td>
<td>+/- 128</td>
<td>+/- 512</td>
<td>+/- 4096</td>
<td>-</td>
</tr>
<tr>
<td>32</td>
<td>+/- 32</td>
<td>+/- 1024</td>
<td>+/- 2048</td>
<td>+/- 32768</td>
<td>+/- 1048576</td>
<td>+/- 33554432</td>
</tr>
<tr>
<td>64</td>
<td>+/- 64</td>
<td>+/- 4096</td>
<td>+/- 8192</td>
<td>+/- 262144</td>
<td>+/- 16777216</td>
<td>+/- 1073741824</td>
</tr>
<tr>
<td>128</td>
<td>+/- 128</td>
<td>+/- 16384</td>
<td>+/- 32768</td>
<td>+/- 2097152</td>
<td>+/- 268435456</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>+/- 256</td>
<td>+/- 65536</td>
<td>+/- 131072</td>
<td>+/- 16777216</td>
<td>Result can overflow on full scale input (&gt; 32-bit signed integer)</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>+/- 1024</td>
<td>+/- 1048576</td>
<td>+/- 2097152</td>
<td>+/- 1073741824</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For more information about Sinc filter type properties and usage, it is recommended to study the theory about digital filters (more resources can be downloaded from internet).

### 33.4.9 Integrator unit

The integrator performs additional decimation and a resolution increase of data coming from the digital filter. The integrator simply performs the sum of data from a digital filter for a given number of data samples from a filter.
The integrator oversampling ratio parameter defines how many data counts will be summed to one data output from the integrator. IOSR can be set in the range 1-256 (see IOSR[7:0] bits description in DFSDM_FLTxFCR register).

### Table 244. Integrator maximum output resolution (peak data values from integrator output) for some IOSR values and FOSR = 256 and Sinc$^5$ filter type (largest data)

<table>
<thead>
<tr>
<th>IOSR</th>
<th>Sinc$^1$</th>
<th>Sinc$^2$</th>
<th>FastSinc</th>
<th>Sinc$^3$</th>
<th>Sinc$^4$</th>
<th>Sinc$^5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>+/- FOSR. x</td>
<td>+/- FOSR$^2$. x</td>
<td>+/- 2 FOSR. x</td>
<td>+/- FOSR$^3$. x</td>
<td>+/- FOSR$^4$. x</td>
<td>+/- FOSR$^5$. x</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+/- 67 108 864</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>32</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+/- 536 870 912</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>128</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+/- 2 147 483</td>
<td>648</td>
<td>-</td>
</tr>
<tr>
<td>256</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+/- 2$^{32}$</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

#### 33.4.10 Analog watchdog

The analog watchdog purpose is to trigger an external signal (break or interrupt) when an analog signal reaches or crosses given maximum and minimum threshold values. An interrupt/event/break generation can then be invoked.

Each analog watchdog will supervise serial data receiver outputs (after the analog watchdog filter on each channel) or data output register (current injected or regular conversion result) according to AWFSEL bit setting (in DFSDM_FLTxCR1 register). The input channels to be monitored or not by the analog watchdog x will be selected by AWDCH[7:0] in DFSDM_FLTxCR2 register.

Analog watchdog conversions on input channels are independent from standard conversions. In this case, the analog watchdog uses its own filters and signal processing on each input channel independently from the main injected or regular conversions. Analog watchdog conversions are performed in a continuous mode on the selected input channels in order to watch channels also when main injected or regular conversions are paused (RCIP = 0, JCIP = 0).

There are high and low threshold registers which are compared with given data values (set by AWHT[23:0] bits in DFSDM_FLTxAWHTR register and by AWLT[23:0] bits in DFSDM_FLTxAWLTR register).
There are 2 options for comparing the threshold registers with the data values

- **Option1**: in this case, the input data are taken from final output data register (AWFSEL=0). This option is characterized by:
  - high input data resolution (up to 24-bits)
  - slow response time - inappropriate for fast response applications like overcurrent detection
  - for the comparison the final data are taken after bit shifting and offset data correction
  - final data are available only after main regular or injected conversions are performed
  - can be used in case of parallel input data source (DATMPX[1:0] ≠ 0 in DFSDM_CHyCFGR1 register)

- **Option2**: in this case, the input data are taken from any serial data receivers output (AWFSEL=1). This option is characterized by:
  - input serial data are processed by dedicated analog watchdog Sinc\(^x\) channel filters with configurable oversampling ratio (1..32) and filter order (1..3) (see AWFOSR[4:0] and AWFORD[1:0] bits setting in DFSDM_CHyAWSCDR register)
  - lower resolution (up to 16-bit)
  - fast response time - appropriate for applications which require a fast response like overcurrent/overvoltage detection)
  - data are available in continuous mode independently from main regular or injected conversions activity

In case of input channels monitoring (AWFSEL=1), the data for comparison to threshold is taken from channels selected by AWDCH[7:0] field (DFSDM_FLTxCR2 register). Each of the selected channels filter result is compared to one threshold value pair (AWHT[23:0] / AWLT[23:0]). In this case, only higher 16 bits (AWHT[23:8] / AWLT[23:8]) define the 16-bit threshold compared with the analog watchdog filter output because data coming from the analog watchdog filter is up to a 16-bit resolution. Bits AWHT[7:0] / AWLT[7:0] are not taken into comparison in this case (AWFSEL=1).

Parameters of the analog watchdog filter configuration for each input channel are set in DFSDM_CHyAWSCDR register (filter order AWFORD[1:0] and filter oversampling ratio AWFOSR[4:0]).

Each input channel has its own comparator which compares the analog watchdog data (from analog watchdog filter) with analog watchdog threshold values (AWHT/AWLT). When several channels are selected (field AWDC[7:0] field of DFSDM_FLTxCR2 register), several comparison requests may be received simultaneously. In this case, the channel request with the lowest number is managed first and then continuing to higher selected channels. For each channel, the result can be recorded in a separate flag (fields AWHFT[7:0], AWTFT[7:0] of DFSDM_FLTxAWSR register). Each channel request is executed in 8 DFSDM clock cycles. So, the bandwidth from each channel is limited to 8 DFSDM clock cycles (if AWDC[7:0] = 0xFF). Because the maximum input channel sampling clock frequency is the DFSDM clock frequency divided by 4, the configuration AWFOSR = 0 (analog watchdog filter is bypassed) cannot be used for analog watchdog feature at this input clock speed. Therefore user must properly configure the number of watched channels and analog watchdog filter parameters with respect to input sampling clock speed and DFSDM frequency.
Analog watchdog filter data for given channel y is available for reading by firmware on field WDATA[15:0] in DFSDM_CHyWDATR register. That analog watchdog filter data is converted continuously (if CHEN=1 in DFSDM_CHyCFGR1 register) with the data rate given by the analog watchdog filter setting and the channel input clock frequency.

The analog watchdog filter conversion works like a regular Fast Continuous Conversion without the integrator. The number of serial samples needed for one result from analog watchdog filter output (at channel input clock frequency fCKIN):

- **first conversion:**
  - for Sincx filters (x=1..5): number of samples = \([\text{FOSR} \times \text{FORD} + \text{FORD} + 1]\)
  - for FastSinc filter: number of samples = \([\text{FOSR} \times 4 + 2 + 1]\)

- **next conversions:**
  - for Sincx and FastSinc filters: number of samples = \([\text{FOSR} \times \text{IOSR}]\)

where:

- \(\text{FOSR}\) ....... filter oversampling ratio: \(\text{FOSR} = \text{AWFOSR}[4:0]+1\) (see DFSDM_CHyAWSCDR register)
- \(\text{FORD}\) ....... the filter order: \(\text{FORD} = \text{AWFORD}[1:0]\) (see DFSDM_CHyAWSCDR register)

In case of output data register monitoring (AWFSEL=0), the comparison is done after a right bit shift and an offset correction of final data (see OFFSET[23:0] and DTRBS[4:0] fields in DFSDM_CHyCFGR2 register). A comparison is performed after each injected or regular end of conversion for the channels selected by AWDCH[7:0] field (in DFSDM_FLTxCR2 register).

The status of an analog watchdog event is signalized in DFSDM_FLTxAWSR register where a given event is latched. AWHTF[y]=1 flag signalizes crossing AWHT[23:0] value on channel y. AWLTF[y]=1 flag signalizes crossing AWLT[23:0] value on channel y. Latched events in DFSDM_FLTxAWSR register are cleared by writing ‘1’ into the corresponding clearing bit CLRRAWHTF[y] or CLRRAWLTF[y] in DFSDM_FLTxAWCFR register.

The global status of an analog watchdog is signalized by the AWDF flag bit in DFSDM_FLTxISR register (it is used for the fast detection of an interrupt source). AWDF=1 signals that at least one watchdog occurred (AWHTF[y]=1 or AWLTF[y]=1 for at least one channel). AWDF bit is cleared when all AWHTF[7:0] and AWLTF[7:0] are cleared.

An analog watchdog event can be assigned to break output signal. There are four break outputs to be assigned to a high or low threshold crossing event (dfsdm_break[3:0]). The break signal assignment to a given analog watchdog event is done by BKAWH[3:0] and BKAWL[3:0] fields in DFSDM_FLTxAWHTR and DFSDM_FLTxAWLTR register.

### 33.4.11 Short-circuit detector

The purpose of a short-circuit detector is to signalize with a very fast response time if an analog signal reached saturated values (out of full scale ranges) and remained on this value given time. This behavior can detect short-circuit or open circuit errors (e.g. overcurrent or overvoltage). An interrupt/event/break generation can be invoked.

Input data into a short-circuit detector is taken from channel transceiver outputs.

There is an upcounting counter on each input channel which is counting consecutive 0’s or 1’s on serial data receiver outputs. A counter is restarted if there is a change in the data stream received - 1 to 0 or 0 to 1 change of data signal. If this counter reaches a short-circuit threshold register value (SCDT[7:0] bits in DFSDM_CHyAWSCDR register), then a short-
Circuit event is invoked. Each input channel has its short-circuit detector. Any channel can be selected to be continuously monitored by setting the SCDEN bit (in DFSDM_CHyCFGR1 register) and it has its own short-circuit detector settings (threshold value in SCDT[7:0] bits, status bit SCDF[7:0], status clearing bits CLRSCDF[7:0]). Status flag SCDF[y] is cleared also by hardware when corresponding channel y is disabled (CHEN[y] = 0).

On each channel, a short-circuit detector event can be assigned to break output signal dfsdm_break[3:0]. There are four break outputs to be assigned to a short-circuit detector event. The break signal assignment to a given channel short-circuit detector event is done by BKSCD[3:0] field in DFSDM_CHyAWSCDR register.

Short circuit detector cannot be used in case of parallel input data channel selection (DATMPX[1:0] ≠ 0 in DFSDM_CHyCFGR1 register).

Four break outputs are totally available (shared with the analog watchdog function).

### 33.4.12 Extreme detector

The purpose of an extremes detector is to collect the minimum and maximum values of final output data words (peak to peak values).

If the output data word is higher than the value stored in the extremes detector maximum register (EXMAX[23:0] bits in DFSDM_FLTxEXMAX register), then this register is updated with the current output data word value and the channel from which the data is stored is in EXMAXCH[2:0] bits (in DFSDM_FLTxEXMAX register).

If the output data word is lower than the value stored in the extremes detector minimum register (EXMIN[23:0] bits in DFSDM_FLTxEXMIN register), then this register is updated with the current output data word value and the channel from which the data is stored is in EXMINCH[2:0] bits (in DFSDM_FLTxEXMIN register).

The minimum and maximum register values can be refreshed by software (by reading given DFSDM_FLTxEXMAX or DFSDM_FLTxEXMIN register). After refresh, the extremes detector minimum data register DFSDM_FLTxEXMIN is filled with 0x7FFFFF (maximum positive value) and the extremes detector maximum register DFSDM_FLTxEXMAX is filled with 0x800000 (minimum negative value).

The extremes detector performs a comparison after a right bit shift and an offset data correction. For each extremes detector, the input channels to be considered into computing the extremes value are selected in EXCH[7:0] bits (in DFSDM_FLTxCR2 register).

### 33.4.13 Data unit block

The data unit block is the last block of the whole processing path: External ΣΔ modulators - Serial transceivers - Sinc filter - Integrator - Data unit block.

The output data rate depends on the serial data stream rate, and filter and integrator settings. The maximum output data rate is:

$$\text{Datarate [samples / s]} = \frac{f_{\text{CKIN}}}{F_{\text{OSR}} \cdot (l_{\text{OSR}} - 1 + F_{\text{ORD}}) + (F_{\text{ORD}} + 1)}$$  \hspace{1cm} \text{...FAST = 0, Sincx filter}

$$\text{Datarate [samples / s]} = \frac{f_{\text{CKIN}}}{F_{\text{OSR}} \cdot (l_{\text{OSR}} - 1 + 4) + (2 + 1)}$$  \hspace{1cm} \text{...FAST = 0, FastSinc filter}
or

\[
\text{Datarate [samples / s]} = \frac{f_{\text{CKIN}}}{F_{\text{OSR}} \cdot I_{\text{OSR}}} \quad \text{...FAST} = 1
\]

Maximum output data rate in case of parallel data input:

\[
\text{Datarate [samples / s]} = \frac{f_{\text{DATAIN RATE}}}{F_{\text{OSR}} \cdot (I_{\text{OSR}} - 2 + F_{\text{ORD}})/\left(F_{\text{ORD}} + 1\right)} \quad \text{...FAST} = 0, \text{Sincx filter}
\]

or

\[
\text{Datarate [samples / s]} = \frac{f_{\text{DATAIN RATE}}}{F_{\text{OSR}} \cdot (I_{\text{OSR}} + 4) + 2/\left(F_{\text{ORD}} + 1\right)} \quad \text{...FAST} = 0, \text{FastSinc filter}
\]

or

\[
\text{Datarate [samples / s]} = \frac{f_{\text{DATAIN RATE}}}{F_{\text{OSR}} \cdot I_{\text{OSR}}} \quad \text{...FAST} = 1 \text{ or any filter bypass case (F}_{\text{OSR}} = 1
\]

where: \(f_{\text{DATAIN RATE}}\) = input data rate from ADC or from CPU/DMA

The right bit-shift of final data is performed in this module because the final data width is 24-bit and data coming from the processing path can be up to 32 bits. This right bit-shift is configurable in the range 0-31 bits for each selected input channel (see DTRBS[4:0] bits in DFSDM.CHyCFGR2 register). The right bit-shift is rounding the result to nearest integer value. The sign of shifted result is maintained - to have valid 24-bit signed format of result data.

In the next step, an offset correction of the result is performed. The offset correction value (OFFSET[23:0] stored in register DFSDM.CHyCFGR2) is subtracted from the output data for a given channel. Data in the OFFSET[23:0] field is set by software by the appropriate calibration routine.

Due to the fact that all operations in digital processing are performed on 32-bit signed registers, the following conditions must be fulfilled not to overflow the result:

\[
\text{FOSR FORD . IOSR} \leq 2^{31} \quad \text{...for Sincx filters, x = 1..5)}
\]

\[
2 \cdot \text{FOSR}^2 \cdot \text{IOSR} \leq 2^{31} \quad \text{...for FastSinc filter}
\]

**Note:** In case of filter and integrator bypass (IOSR[7:0]=0, FOSR[9:0]=0), the input data rate \(f_{\text{DATAIN RATE}}\) must be limited to be able to read all output data:

\[
f_{\text{DATAIN RATE}} \leq f_{\text{APB}}
\]

where \(f_{\text{APB}}\) is the bus frequency to which the DFSDM peripheral is connected.

### 33.4.14 Signed data format

Each DFSDM input serial channel can be connected to one external ΣΔ modulator. An external ΣΔ modulator can have 2 differential inputs (positive and negative) which can be used for a differential or single-ended signal measurement.

A ΣΔ modulator output is always assumed in a signed format (a data stream of zeros and ones from a ΣΔ modulator represents values -1 and +1).
Signed data format in registers: Data is in a signed format in registers for final output data, analog watchdog, extremes detector, offset correction. The msb of output data word represents the sign of value (two’s complement format).

### 33.4.15 Launching conversions

**Injected conversions** can be launched using the following methods:

- **Software**: writing ‘1’ to JSWSTART in the DFSDM_FLT\textsubscript{x}CR1 register.
- **Trigger**: JEXTSE\textsubscript{L}[4:0] selects the trigger signal while JEXTEN activates the trigger and selects the active edge at the same time (see the DFSDM_FLT\textsubscript{x}CR1 register).
- **Synchronous with DFSDM_FLT\textsubscript{0} if JSYNC=1**: for DFSDM_FLT\textsubscript{x} (x>0), an injected conversion is automatically launched when in DFSDM_FLT\textsubscript{0}; the injected conversion is started by software (JSWSTART=1 in DFSDM_FLT\textsubscript{0}CR2 register). Each injected conversion in DFSDM_FLT\textsubscript{x} (x>0) is always executed according to its local configuration settings (JSCAN, JCHG, etc.).

If the scan conversion is enabled (bit JSCAN=1) then, each time an injected conversion is triggered, all of the selected channels in the injected group (JCHG[7:0] bits in DFSDM_FLT\textsubscript{x}JCHGR register) are converted sequentially, starting with the lowest channel (channel 0, if selected).

If the scan conversion is disabled (bit JSCAN=0) then, each time an injected conversion is triggered, only one of the selected channels in the injected group (JCHG[7:0] bits in DFSDM_FLT\textsubscript{x}JCHGR register) is converted and the channel selection is then moved to the next selected channel. Writing to the JCHG[7:0] bits when JSCAN=0 sets the channel selection to the lowest selected injected channel.

Only one injected conversion can be ongoing at a given time. Thus, any request to launch an injected conversion is ignored if another request for an injected conversion has already been issued but not yet completed.

**Regular conversions** can be launched using the following methods:

- **Software**: by writing ‘1’ to RSWSTART in the DFSDM_FLT\textsubscript{x}CR1 register.
- **Synchronous with DFSDM_FLT\textsubscript{0} if RSYNC=1**: for DFSDM_FLT\textsubscript{x} (x>0), a regular conversion is automatically launched when in DFSDM_FLT\textsubscript{0}; a regular conversion is started by software (RSWSTART=1 in DFSDM_FLT\textsubscript{0}CR2 register). Each regular conversion in DFSDM_FLT\textsubscript{x} (x>0) is always executed according to its local configuration settings (RCONT, RCH, etc.).

Only one regular conversion can be pending or ongoing at a given time. Thus, any request to launch a regular conversion is ignored if another request for a regular conversion has already been issued but not yet completed. A regular conversion can be pending if it was interrupted by an injected conversion or if it was started while an injected conversion was in progress. This pending regular conversion is then delayed and is performed when all injected conversion are finished. Any delayed regular conversion is signalized by RPEND bit in DFSDM_FLT\textsubscript{x}RDATAR register.

### 33.4.16 Continuous and fast continuous modes

Setting RCONT in the DFSDM_FLT\textsubscript{x}CR1 register causes regular conversions to execute in continuous mode. RCONT=1 means that the channel selected by RCH[2:0] is converted repeatedly after ‘1’ is written to RSWSTART.
The regular conversions executing in continuous mode can be stopped by writing ‘0’ to RCONT. After clearing RCONT, the on-going conversion is stopped immediately.

In continuous mode, the data rate can be increased by setting the FAST bit in the DFSDM_FLTxCR1 register. In this case, the filter does not need to be refilled by new fresh data if converting continuously from one channel because data inside the filter is valid from previously sampled continuous data. The speed increase depends on the chosen filter order. The first conversion in fast mode (FAST=1) after starting a continuous conversion by RSWSTART=1 takes still full time (as when FAST=0), then each subsequent conversion is finished in shorter intervals.

Conversion time in continuous mode:

if FAST = 0 (or first conversion if FAST=1):
  for Sincx filters:
    \[ t = \frac{\text{CNVCNT}}{f_{\text{DFSDMCLK}}} = \frac{[F_{\text{OSR}} \cdot (I_{\text{OSR}}-1 + F_{\text{ORD}}) + F_{\text{ORD}}]}{f_{\text{CKIN}}} \]
  for FastSinc filter:
    \[ t = \frac{\text{CNVCNT}}{f_{\text{DFSDMCLK}}} = \frac{[F_{\text{OSR}} \cdot (I_{\text{OSR}}-1 + 4) + 2]}{f_{\text{CKIN}}} \]

if FAST = 1 (except first conversion):
  for Sincx and FastSinc filters:
    \[ t = \frac{\text{CNVCNT}}{f_{\text{DFSDMCLK}}} = \frac{[F_{\text{OSR}} \cdot I_{\text{OSR}}]}{f_{\text{CKIN}}} \]
  in case F_{\text{OSR}} = FOSR[9:0]+1 = 1 (filter bypassed, only integrator active):
    \[ t = \frac{I_{\text{OSR}}}{f_{\text{CKIN}}} \] (… but CNVCNT=0)

Continuous mode is not available for injected conversions. Injected conversions can be started by timer trigger to emulate the continuous mode with precise timing.

If a regular continuous conversion is in progress (RCONT=1) and if a write access to DFSDM_FLTxCR1 register requesting regular continuous conversion (RCONT=1) is performed, then regular continuous conversion is restarted from the next conversion cycle (like new regular continuous conversion is applied for new channel selection - even if there is no change in DFSDM_FLTxCR1 register).

### 33.4.17 Request precedence

An injected conversion has a higher precedence than a regular conversion. A regular conversion which is already in progress is immediately interrupted by the request of an injected conversion; this regular conversion is restarted after the injected conversion finishes.

An injected conversion cannot be launched if another injected conversion is pending or already in progress: any request to launch an injected conversion (either by JSWSTART or by a trigger) is ignored as long as bit JCIP is ‘1’ (in the DFSDM_FLTxISR register).

Similarly, a regular conversion cannot be launched if another regular conversion is pending or already in progress: any request to launch a regular conversion (using RSWSTART) is ignored as long as bit RCIP is ‘1’ (in the DFSDM_FLTxISR register).

However, if an injected conversion is requested while a regular conversion is already in progress, the regular conversion is immediately stopped and an injected conversion is launched. The regular conversion is then restarted and this delayed restart is signalized in bit RPEND.

Injected conversions have precedence over regular conversions in that a injected conversion can temporarily interrupt a sequence of continuous regular conversions. When
the sequence of injected conversions finishes, the continuous regular conversions start again if RCONT is still set (and RPEND bit will signalize the delayed start on the first regular conversion result).

Precedence also matters when actions are initiated by the same write to DFSDM, or if multiple actions are pending at the end of another action. For example, suppose that, while an injected conversion is in process (JCIP=1), a single write operation to DFSDM_FLTxCR1 writes ‘1’ to RSWSTART, requesting a regular conversion. When the injected sequence finishes, the precedence dictates that the regular conversion is performed next and its delayed start is signalized in RPEND bit.

33.4.18 Power optimization in run mode

In order to reduce the consumption, the DFSDM filter and integrator are automatically put into idle when not used by conversions (RCIP=0, JCIP=0).

33.5 DFSDM interrupts

In order to increase the CPU performance, a set of interrupts related to the CPU event occurrence has been implemented:

- End of injected conversion interrupt:
  - enabled by JEOCIE bit in DFSDM_FLTxCR2 register
  - indicated in JEOCF bit in DFSDM_FLTxISR register
  - cleared by reading DFSDM_FLTxJDATAR register (injected data)
  - indication of which channel end of conversion occurred, reported in JDATACH[2:0] bits in DFSDM_FLTxJDATAR register

- End of regular conversion interrupt:
  - enabled by REOCIE bit in DFSDM_FLTxCR2 register
  - indicated in REOCF bit in DFSDM_FLTxISR register
  - cleared by reading DFSDM_FLTxRDATAR register (regular data)
  - indication of which channel end of conversion occurred, reported in RDATACH[2:0] bits in DFSDM_FLTxRDATAR register

- Data overrun interrupt for injected conversions:
  - occurred when injected converted data were not read from DFSDM_FLTxJDATAR register (by CPU or DMA) and were overwritten by a new injected conversion
  - enabled by JOVRIE bit in DFSDM_FLTxCR2 register
  - indicated in JOVRF bit in DFSDM_FLTxISR register
  - cleared by writing ‘1’ into CLRJOVRF bit in DFSDM_FLTxICR register

- Data overrun interrupt for regular conversions:
  - occurred when regular converted data were not read from DFSDM_FLTxRDATAR register (by CPU or DMA) and were overwritten by a new regular conversion
  - enabled by ROVRIE bit in DFSDM_FLTxCR2 register
  - indicated in ROVRF bit in DFSDM_FLTxISR register
  - cleared by writing ‘1’ into CLRROVRF bit in DFSDM_FLTxICR register

- Analog watchdog interrupt:
– occurred when converted data (output data or data from analog watchdog filter - according to AWFSEL bit setting in DFSDM_FLTxCR1 register) crosses over/under high/low thresholds in DFSDM_FLTxAWHTR / DFSDM_FLTxAWLTR registers
– enabled by AWDie bit in DFSDM_FLTxCR2 register (on selected channels AWDCH[7:0])
– indicated in AWDF bit in DFSDM_FLTxISR register
– separate indication of high or low analog watchdog threshold error by AWHTF[7:0] and AWLTF[7:0] fields in DFSDM_FLTxAWSR register
– cleared by writing ‘1’ into corresponding CLRAWHTF[7:0] or CLRAWLTF[7:0] bits in DFSDM_FLTxAWCFR register

• Short-circuit detector interrupt:
  – occurred when the number of stable data crosses over thresholds in DFSDM_CHyAWSCDR register
  – enabled by SCDIE bit in DFSDM_FLTxCR2 register (on channel selected by SCDEN bit in DFSDM_CHyCFGR1 register)
  – indicated in SCDF[7:0] bits in DFSDM_FLTxISR register (which also reports the channel on which the short-circuit detector event occurred)
  – cleared by writing ‘1’ into the corresponding CLRSCDF[7:0] bit in DFSDM_FLTxICR register

• Channel clock absence interrupt:
  – occurred when there is clock absence on CKINy pin (see Clock absence detection in Section 33.4.4: Serial channel transceivers)
  – enabled by CKABIE bit in DFSDM_FLTxCR2 register (on channels selected by CKABEN bit in DFSDM_CHyCFGR1 register)
  – indicated in CKABF[y] bit in DFSDM_FLTxISR register
  – cleared by writing ‘1’ into CLRCKABF[y] bit in DFSDM_FLTxICR register

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Event/Interrupt clearing method</th>
<th>Interrupt enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>End of injected conversion</td>
<td>JEOCF</td>
<td>reading DFSDM_FLTxJDATAR</td>
<td>JEOCIE</td>
</tr>
<tr>
<td>End of regular conversion</td>
<td>REOCF</td>
<td>reading DFSDM_FLTxRDATAR</td>
<td>REOCIE</td>
</tr>
<tr>
<td>Injected data overrun</td>
<td>JOVRF</td>
<td>writing CLRJOVRF = 1</td>
<td>JOVRIE</td>
</tr>
<tr>
<td>Regular data overrun</td>
<td>ROVRF</td>
<td>writing CLRROVRF = 1</td>
<td>ROVRIE</td>
</tr>
<tr>
<td>Analog watchdog</td>
<td>AWDF, AWHTF[7:0], AWLTF[7:0]</td>
<td>writing CLRAWHTF[7:0] = 1, writing CLRAWLTF[7:0] = 1</td>
<td>AWDIE, (AWDCH[7:0])</td>
</tr>
<tr>
<td>short-circuit detector</td>
<td>SCDF[7:0]</td>
<td>writing CLRSCDF[7:0] = 1</td>
<td>SCDIE, (SCDEN)</td>
</tr>
<tr>
<td>Channel clock absence</td>
<td>CKABF[7:0]</td>
<td>writing CLRCKABF[7:0] = 1</td>
<td>CKABIE, (CKABEN)</td>
</tr>
</tbody>
</table>
33.6 DFSDM DMA transfer

To decrease the CPU intervention, conversions can be transferred into memory using a DMA transfer. A DMA transfer for injected conversions is enabled by setting bit JDMAEN=1 in DFSDM_FLTxCR1 register. A DMA transfer for regular conversions is enabled by setting bit RDMAEN=1 in DFSDM_FLTxCR1 register.

Note: With a DMA transfer, the interrupt flag is automatically cleared at the end of the injected or regular conversion (JEOCF or REOCF bit in DFSDM_FLTxISR register) because DMA is reading DFSDM_FLTxJDATAR or DFSDM_FLTxRDATAR register.

33.7 DFSDM channel y registers (y=0..7)

Word access (32-bit) must be used for registers write access except DFSDM_CHyDATINR register. Write access to DFSDM_CHyDATINR register can be either word access (32-bit) or half-word access (16-bit).

33.7.1 DFSDM channel y configuration register (DFSDM_CHyCFGR1)

This register specifies the parameters used by channel y.

Address offset: 0x00 + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>DFSDMEN: Global enable for DFSDM interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>DFSDM interface disabled</td>
</tr>
<tr>
<td>1:</td>
<td>DFSDM interface enabled</td>
</tr>
<tr>
<td></td>
<td>If DFSDM interface is enabled, then it is</td>
</tr>
<tr>
<td></td>
<td>started to operate according to enabled y</td>
</tr>
<tr>
<td></td>
<td>channels and enabled x filters settings</td>
</tr>
<tr>
<td></td>
<td>(CHEN bit in DFSDM_CHyCFGR1 and DFEN bit in</td>
</tr>
<tr>
<td></td>
<td>DFSDM_FLTxCR1).</td>
</tr>
<tr>
<td></td>
<td>Data cleared by setting DFSDMEN=0:</td>
</tr>
<tr>
<td></td>
<td>–all registers DFSDM_FLTxISR are set to</td>
</tr>
<tr>
<td></td>
<td>reset state (x = 0..7)</td>
</tr>
<tr>
<td></td>
<td>–all registers DFSDM_FLTxAWSR are set to</td>
</tr>
<tr>
<td></td>
<td>reset state (x = 0..7)</td>
</tr>
<tr>
<td></td>
<td>Note: DFSDMEN is present only in DFSDM_Ch0CFGR1 register (channel y=0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>CKOUTSRC: Output serial clock source selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>Source for output clock is from system clock</td>
</tr>
<tr>
<td>1:</td>
<td>Source for output clock is from audio clock</td>
</tr>
<tr>
<td></td>
<td>This value can be modified only when</td>
</tr>
<tr>
<td></td>
<td>DFSDMEN=0 (in DFSDM_CH0CFGR1 register).</td>
</tr>
<tr>
<td></td>
<td>Note: CKOUTSRC is present only in DFSDM_Ch0CFGR1 register (channel y=0)</td>
</tr>
</tbody>
</table>

Bits 29:24 Reserved, must be kept at reset value.
Bits 23:16 **CKOUTDIV[7:0]**: Output serial clock divider
0: Output clock generation is disabled (CKOUT signal is set to low state)
1-255: Defines the division of system clock for the serial clock output for CKOUT signal in range 2-256 (Divider = CKOUTDIV+1).
CKOUTDIV also defines the threshold for a clock absence detection.
This value can only be modified when DFSDMEN=0 (in DFSDM_CH0CFGR1 register).
If DFSDMEN=0 (in DFSDM_CH0CFGR1 register) then CKOUT signal is set to low state (setting is performed one DFSDM clock cycle after DFSDMEN=0).
*Note: CKOUTDIV is present only in DFSDM_CH0CFGR1 register (channel y=0)*

Bits 15:14 **DATPACK[1:0]**: Data packing mode in DFSDM_CHyDATINR register.
0: Standard: input data in DFSDM_CHyDATINR register are stored only in INDAT0[15:0]. To empty DFSDM_CHyDATINR register one sample must be read by the DFSDM filter from channel y.
1: Interleaved: input data in DFSDM_CHyDATINR register are stored as two samples:
–first sample in INDAT0[15:0] (assigned to channel y)
–second sample INDAT1[15:0] (assigned to channel y)
To empty DFSDM_CHyDATINR register, two samples must be read by the digital filter from channel y (INDAT0[15:0] part is read as first sample and then INDAT1[15:0] part is read as next sample).
2: Dual: input data in DFSDM_CHyDATINR register are stored as two samples:
–first sample INDAT0[15:0] (assigned to channel y)
–second sample INDAT1[15:0] (assigned to channel y+1)
To empty DFSDM_CHyDATINR register first sample must be read by the digital filter from channel y and second sample must be read by another digital filter from channel y+1. Dual mode is available only on even channel numbers (y = 0, 2, 4, 6), for odd channel numbers (y = 1, 3, 5, 7) DFSDM_CHyDATINR is write protected. If an even channel is set to dual mode then the following odd channel must be set into standard mode (DATPACK[1:0]=0) for correct cooperation with even channel.
3: Reserved
This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 13:12 **DATMPX[1:0]**: Input data multiplexer for channel y
0: Data to channel y are taken from external serial inputs as 1-bit values. DFSDM_CHyDATINR register is write protected.
1: Data to channel y are taken from internal analog to digital converter ADCy+1 output register update as 16-bit values (if ADCy+1 is available). Data from ADCs are written into INDAT0[15:0] part of DFSDM_CHyDATINR register.
2: Data to channel y are taken from internal DFSDM_CHyDATINR register by direct CPU/DMA write. There can be written one or two 16-bit data samples according DATPACK[1:0] bit field setting.
3: Reserved
This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **CHINSEL**: Channel inputs selection
0: Channel inputs are taken from pins of the same channel y.
1: Channel inputs are taken from pins of the following channel (channel (y+1) modulo 8).
This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bit 7 **CHEN**: Channel y enable
0: Channel y disabled
1: Channel y enabled
If channel y is enabled, then serial data receiving is started according to the given channel setting.
Bit 6 **CKABEN**: Clock absence detector enable on channel y
- 0: Clock absence detector disabled on channel y
- 1: Clock absence detector enabled on channel y

Bit 5 **SCDEN**: Short-circuit detector enable on channel y
- 0: Input channel y will not be guarded by the short-circuit detector
- 1: Input channel y will be continuously guarded by the short-circuit detector

Bit 4 Reserved, must be kept at reset value.

**Bits 3:2** **SPICKSEL[1:0]**: SPI clock select for channel y
- 0: clock coming from external CKINy input - sampling point according SITP[1:0]
- 1: clock coming from internal CKOUT output - sampling point according SITP[1:0]
- 2: clock coming from internal CKOUT - sampling point on each second CKOUT falling edge.
  
  For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input rising edge).
- 3: clock coming from internal CKOUT output - sampling point on each second CKOUT rising edge.
  
  For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input falling edge).

This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

**Bits 1:0** **SITP[1:0]**: Serial interface type for channel y
- 00: SPI with rising edge to strobe data
- 01: SPI with falling edge to strobe data
- 10: Manchester coded input on DATINy pin: rising edge = logic 0, falling edge = logic 1
- 11: Manchester coded input on DATINy pin: rising edge = logic 1, falling edge = logic 0

This value can only be modified when CHEN=0 (in DFSDM_CHyCFGR1 register).

### 33.7.2 DFSDM channel y configuration register (DFSDM_CHyCFGR2)

This register specifies the parameters used by channel y.

Address offset: 0x04 + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

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

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

---

**OFFSET[7:0]**

**DTRBS[4:0]**

**Res. Res. Res.**
Bits 31:8 **OFFSET[23:0]**: 24-bit calibration offset for channel y
For channel y, OFFSET is applied to the results of each conversion from this channel. This value is set by software.

Bits 7:3 **DTRBS[4:0]**: Data right bit-shift for channel y
0-31: Defines the shift of the data result coming from the integrator - how many bit shifts to the right will be performed to have final results. Bit-shift is performed before offset correction. The data shift is rounding the result to nearest integer value. The sign of shifted result is maintained (to have valid 24-bit signed format of result data).
This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 2:0 Reserved, must be kept at reset value.

### 33.7.3 DFSDM channel y analog watchdog and short-circuit detector register (DFSDM_CHyAWSCDR)
Short-circuit detector and analog watchdog settings for channel y.
Address offset: 0x08 + 0x20 * y, (y = 0 to 7)
Reset value: 0x0000 0000

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **AWFORD[1:0]**: Analog watchdog Sinc filter order on channel y
0: FastSinc filter type
1: Sinc1 filter type
2: Sinc2 filter type
3: Sinc3 filter type
Sincx filter type transfer function:

\[ H(z) = \left( \frac{1 - z^{-FOSR}}{1 - z^{-1}} \right)^x \]

FastSinc filter type transfer function:

\[ H(z) = \left( \frac{1 - z^{-FOSR}}{1 - z^{-1}} \right)^2 \cdot (1 + z^{-2 \cdot FOSR}) \]

This bit can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **AWFOSR[4:0]**: Analog watchdog filter oversampling ratio (decimation rate) on channel y
0 - 31: Defines the length of the Sinc type filter in the range 1 - 32 (AWFOSR + 1). This number is also the decimation ratio of the analog data rate.
This bit can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

*Note:* If AWFOSR = 0 then the filter has no effect (filter bypass).
33.7.4 DFSDM channel y watchdog filter data register (DFSDM_CHyWDATR)

This register contains the data resulting from the analog watchdog filter associated to the input channel y.

Address offset: 0x0C + 0x20 * y, (y = 0 to 7)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 15:12 BKSCD[3:0]: Break signal assignment for short-circuit detector on channel y</th>
</tr>
</thead>
<tbody>
<tr>
<td>BKSCD[i] = 0: Break i signal not assigned to short-circuit detector on channel y</td>
</tr>
<tr>
<td>BKSCD[i] = 1: Break i signal assigned to short-circuit detector on channel y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 7:0 SCDT[7:0]: short-circuit detector threshold for channel y</th>
</tr>
</thead>
<tbody>
<tr>
<td>These bits are written by software to define the threshold counter for the short-circuit detector. If this value is reached, then a short-circuit detector event occurs on a given channel.</td>
</tr>
</tbody>
</table>

33.7.5 DFSDM channel y data input register (DFSDM_CHyDATINR)

This register contains 16-bit input data to be processed by DFSDM filter module. Write access can be either word access (32-bit) or half-word access (16-bit).

Address offset: 0x10 + 0x20 * y, (y = 0 to 7)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 15:0 WDATA[15:0]: Input channel y watchdog data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data converted by the analog watchdog filter for input channel y. This data is continuously converted (no trigger) for this channel, with a limited resolution (OSR=1..32/sinc order = 1..3).</td>
</tr>
</tbody>
</table>
31:16 **INDAT1[15:0]**: Input data for channel y or channel y+1
- Input parallel channel data to be processed by the digital filter if DATMPX[1:0]=1 or DATMPX[1:0]=2.
- Data can be written by CPU/DMA (if DATMPX[1:0]=2) or directly by internal ADC (if DATMPX[1:0]=1).
- If DATPACK[1:0]=0 (standard mode)
  - INDAT0[15:0] is write protected (not used for input sample).
- If DATPACK[1:0]=1 (interleaved mode)
  - Second channel y data sample is stored into INDAT1[15:0]. First channel y data sample is stored into INDAT0[15:0]. Both samples are read sequentially by DFSDM_FLTx filter as two channel y data samples.
  - If DATPACK[1:0]=2 (dual mode).
    - For even y channels: sample in INDAT1[15:0] is automatically copied into INDAT0[15:0] of channel (y+1).
    - For odd y channels: INDAT1[15:0] is write protected.
  - See Section 33.4.6: Parallel data inputs for more details.

Bits 15:0 **INDAT0[15:0]**: Input data for channel y
- Input parallel channel data to be processed by the digital filter if DATMPX[1:0]=1 or DATMPX[1:0]=2.
- Data can be written by CPU/DMA (if DATMPX[1:0]=2) or directly by internal ADC (if DATMPX[1:0]=1).
- If DATPACK[1:0]=0 (standard mode)
  - Channel y data sample is stored into INDAT0[15:0].
- If DATPACK[1:0]=1 (interleaved mode)
  - First channel y data sample is stored into INDAT0[15:0]. Second channel y data sample is stored into INDAT1[15:0]. Both samples are read sequentially by DFSDM_FLTx filter as two channel y data samples.
  - If DATPACK[1:0]=2 (dual mode).
    - For even y channels: Channel y data sample is stored into INDAT0[15:0].
    - For odd y channels: INDAT0[15:0] is write protected.
  - See Section 33.4.6: Parallel data inputs for more details.

**33.7.6 DFSDM channel y delay register (DFSDM_CHyDLYR)**

Address offset: 0x14 + 0x20 * y, (y = 0 to 7)
Reset value: 0x0000 0000

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

33.4.6 Parallel data inputs
33.8 DFSDM filter x module registers (x=0..7)

Word access (32-bit) must be used for registers write access except DFSDM_CHyDATINR register.

33.8.1 DFSDM filter x control register 1 (DFSDM_FLTxCR1)

Address offset: 0x100 + 0x80 * x, (x = 0 to 7)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rt_w1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rt_w1</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bit 30 AWFSEL: Analog watchdog fast mode select
0: Analog watchdog on data output value (after the digital filter). The comparison is done after offset correction and shift
1: Analog watchdog on channel transceivers value (after watchdog filter)

Bit 29 FAST: Fast conversion mode selection for regular conversions
0: Fast conversion mode disabled
1: Fast conversion mode enabled

When converting a regular conversion in continuous mode, having enabled the fast mode causes each conversion (except the first) to execute faster than in standard mode. This bit has no effect on conversions which are not continuous.

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

If FAST=0 (or first conversion in continuous mode if FAST=1):
\[ t = \left[ \frac{FOSR}{IOSR-1 + FORD} + FORD \right] / f_{CKIN} \] ....... for Sinc^x filters
\[ t = \left[ \frac{FOSR}{IOSR-1 + 4} + 2 \right] / f_{CKIN} \] ....... for FastSinc filter

If FAST=1 in continuous mode (except first conversion):
\[ t = \left[ FOSR - IOSR \right] / f_{CKIN} \]

in case if \( FOSR = FOSR[9:0]+1 = 1 \) (filter bypassed, active only integrator):
\[ t = I_{OSR} / f_{CKIN} \] (but CNVCNT=0)

where: \( f_{CKIN} \) is the channel input clock frequency (on given channel CKINy pin) or input data rate in case of parallel data input.

Bits 28:27 Reserved, must be kept at reset value.
Bits 26:24 **RCH[2:0]**: Regular channel selection
- 0: Channel 0 is selected as the regular channel
- 1: Channel 1 is selected as the regular channel
- ...
- 7: Channel 7 is selected as the regular channel
Writing these bits when RCIP=1 takes effect when the next regular conversion begins. This is especially useful in continuous mode (when RCONT=1). It also affects regular conversions which are pending (due to ongoing injected conversion).

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **RDMAEN**: DMA channel enabled to read data for the regular conversion
- 0: The DMA channel is not enabled to read regular data
- 1: The DMA channel is enabled to read regular data
  This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RSYNC**: Launch regular conversion synchronously with DFSDM_FLT0
- 0: Do not launch a regular conversion synchronously with DFSDM_FLT0
- 1: Launch a regular conversion in this DFSDM_FLTx at the very moment when a regular conversion is launched in DFSDM_FLT0
  This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 18 **RCONT**: Continuous mode selection for regular conversions
- 0: The regular channel is converted just once for each conversion request
- 1: The regular channel is converted repeatedly after each conversion request
  Writing ‘0’ to this bit while a continuous regular conversion is already in progress stops the continuous mode immediately.

Bit 17 **RSWSTART**: Software start of a conversion on the regular channel
- 0: Writing ‘0’ has no effect
- 1: Writing ‘1’ makes a request to start a conversion on the regular channel and causes RCIP to become ‘1’. If RCIP=1 already, writing to RSWSTART has no effect. Writing ‘1’ has no effect if RSYNC=1.
  This bit is always read as ‘0’.

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:13 **JEXTEN[1:0]**: Trigger enable and trigger edge selection for injected conversions
- 00: Trigger detection is disabled
- 01: Each rising edge on the selected trigger makes a request to launch an injected conversion
- 10: Each falling edge on the selected trigger makes a request to launch an injected conversion
- 11: Both rising edges and falling edges on the selected trigger make requests to launch injected conversions
  This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).
Bits 12:8 **JEXTSEL[4:0]**: Trigger signal selection for launching injected conversions
- 0x0-0x1F: Trigger inputs selected by the following table (internal or external trigger).
  - This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).
  - Note: synchronous trigger has latency up to one $f_{DFSDMCLK}$ clock cycle (with deterministic jitter), asynchronous trigger has latency 2-3 $f_{DFSDMCLK}$ clock cycles (with jitter up to 1 cycle).

<table>
<thead>
<tr>
<th>DFSDM_FLTx</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>dfsdm_jtrg0</td>
</tr>
<tr>
<td>0x01</td>
<td>dfsdm_jtrg1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0x1E</td>
<td>dfsdm_jtrg30</td>
</tr>
<tr>
<td>0x1F</td>
<td>dfsdm_jtrg31</td>
</tr>
</tbody>
</table>
  - Refer to Table 241: DFSDM triggers connection.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **JDMAEN**: DMA channel enabled to read data for the injected channel group
- 0: The DMA channel is not enabled to read injected data
- 1: The DMA channel is enabled to read injected data
  - This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 4 **JSCAN**: Scanning conversion mode for injected conversions
- 0: One channel conversion is performed from the injected channel group and next the selected channel from this group is selected.
- 1: The series of conversions for the injected group channels is executed, starting over with the lowest selected channel.
  - This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).
  - Writing JCHG if JSCAN=0 resets the channel selection to the lowest selected channel.

Bit 3 **JSYNC**: Launch an injected conversion synchronously with the DFSDM_FLT0 JSWSTART trigger
- 0: Do not launch an injected conversion synchronously with DFSDM_FLT0
- 1: Launch an injected conversion in this DFSDM_FLTx at the very moment when an injected conversion is launched in DFSDM_FLT0 by its JSWSTART trigger
  - This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **JSWSTART**: Start a conversion of the injected group of channels
- 0: Writing ‘0’ has no effect.
- 1: Writing ‘1’ makes a request to convert the channels in the injected conversion group, causing JCIP to become ‘1’ at the same time. If JCIP=1 already, then writing to JSWSTART has no effect. Writing ‘1’ has no effect if JSYNC=1.
  - This bit is always read as ‘0’.

Bit 0 **DFEN**: DFSDM_FLTx enable
- 0: DFSDM_FLTx is disabled. All conversions of given DFSDM_FLTx are stopped immediately and all DFSDM_FLTx functions are stopped.
- 1: DFSDM_FLTx is enabled. If DFSDM_FLTx is enabled, then DFSDM_FLTx starts operating according to its setting.
  - Data which are cleared by setting DFEN=0:
    - register DFSDM_FLTxISR is set to the reset state
    - register DFSDM_FLTxAWSR is set to the reset state
33.8.2 DFSDM filter x control register 2 (DFSDM_FLTxCR2)

Address offset: 0x104 + 0x80 * x, (x = 0 to 7)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>EXCH[7:0]</td>
<td>Res.</td>
<td>CKABIE</td>
<td>SCDIE</td>
<td>AWDIE</td>
<td>ROVRIE</td>
<td>JOVRIE</td>
<td>REOCIE</td>
<td>JEOCIIE</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **AWDCH[7:0]**: Analog watchdog channel selection
These bits select the input channel to be guarded continuously by the analog watchdog.
**AWDCH[y] = 0**: Analog watchdog is disabled on channel y  
**AWDCH[y] = 1**: Analog watchdog is enabled on channel y

Bits 15:8 **EXCH[7:0]**: Extremes detector channel selection
These bits select the input channels to be taken by the Extremes detector.
**EXCH[y] = 0**: Extremes detector does not accept data from channel y  
**EXCH[y] = 1**: Extremes detector accepts data from channel y

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CKABIE**: Clock absence interrupt enable
0: Detection of channel input clock absence interrupt is disabled  
1: Detection of channel input clock absence interrupt is enabled
Please see the explanation of CKABF[7:0] in DFSDM_FLTxISR.

*Note: CKABIE is present only in DFSDM_FLT0CR2 register (filter x=0)*

Bit 5 **SCDIE**: Short-circuit detector interrupt enable
0: short-circuit detector interrupt is disabled  
1: short-circuit detector interrupt is enabled
Please see the explanation of SCDF[7:0] in DFSDM_FLTxISR.

*Note: SCDIE is present only in DFSDM_FLT0CR2 register (filter x=0)*

Bit 4 **AWDIE**: Analog watchdog interrupt enable
0: Analog watchdog interrupt is disabled  
1: Analog watchdog interrupt is enabled
Please see the explanation of AWDF in DFSDM_FLTxISR.

Bit 3 **ROVRIE**: Regular data overrun interrupt enable
0: Regular data overrun interrupt is disabled  
1: Regular data overrun interrupt is enabled
Please see the explanation of ROVRF in DFSDM_FLTxISR.
Bit 2  **JOVRIE**: Injected data overrun interrupt enable
   0: Injected data overrun interrupt is disabled
   1: Injected data overrun interrupt is enabled
   Please see the explanation of JOVRF in DFSDM_FLTxISR.

Bit 1  **REOCE**: Regular end of conversion interrupt enable
   0: Regular end of conversion interrupt is disabled
   1: Regular end of conversion interrupt is enabled
   Please see the explanation of REOCF in DFSDM_FLTxISR.

Bit 0  **JEOCIE**: Injected end of conversion interrupt enable
   0: Injected end of conversion interrupt is disabled
   1: Injected end of conversion interrupt is enabled
   Please see the explanation of JEOCF in DFSDM_FLTxISR.

### 33.8.3 DFSDM filter x interrupt and status register (DFSDM_FLTxISR)

Address offset: $0x108 + 0x80 \times x$, ($x = 0$ to 7)
Reset value: $0x00FF\ 0000$

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>SCDF[7:0] (short-circuit detector flag)</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>CKABF[7:0] (clock absence flag)</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>RCIP (regular conversion in progress status)</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>JCIP (regular conversion in progress status)</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Res. (regular conversion in progress status)</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Res. (regular conversion in progress status)</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Res. (regular conversion in progress status)</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Res. (regular conversion in progress status)</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Res. (regular conversion in progress status)</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>AWDF (regular conversion in progress status)</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>ROVRF (regular conversion in progress status)</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>JOVRF (regular conversion in progress status)</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>REOCF (regular conversion in progress status)</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>JEOCF (regular conversion in progress status)</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Res. (regular conversion in progress status)</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Res. (regular conversion in progress status)</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24  **SCDF[7:0]**: short-circuit detector flag
   SCDF[y]=0: No short-circuit detector event occurred on channel y
   SCDF[y]=1: The short-circuit detector counter reaches, on channel y, the value programmed in the DFSDM_CHyAWSCDR registers
   This bit is set by hardware. It can be cleared by software using the corresponding CLRSCDF[y] bit in the DFSDM_FLTxICR register. SCDF[y] is cleared also by hardware when CHEN[y] = 0 (given channel is disabled).
   Note: **SCDF[7:0] is present only in DFSDM_FLT0ISR register (filter x=0)**

Bits 23:16  **CKABF[7:0]**: Clock absence flag
   CKABF[y]=0: Clock signal on channel y is present.
   CKABF[y]=1: Clock signal on channel y is not present.
   Given y bit is set by hardware when clock absence is detected on channel y. It is held at CKABF[y]=1 state by hardware when CHEN=0 (see DFSDM_CHyCFG1 register). It is held at CKABF[y]=1 state by hardware when the transceiver is not yet synchronized. It can be cleared by software using the corresponding CLRCKABF[y] bit in the DFSDM_FLTxICR register.
   Note: **CKABF[7:0] is present only in DFSDM_FLT0ISR register (filter x=0)**

Bit 15  Reserved, must be kept at reset value.

Bit 14  **RCIP**: Regular conversion in progress status
   0: No request to convert the regular channel has been issued
   1: The conversion of the regular channel is in progress or a request for a regular conversion is pending
   A request to start a regular conversion is ignored when RCIP=1.
Bit 13  **JCIP**: Injected conversion in progress status  
0: No request to convert the injected channel group (neither by software nor by trigger) has been issued  
1: The conversion of the injected channel group is in progress or a request for a injected conversion is pending, due either to ‘1’ being written to JSWSTART or to a trigger detection  
A request to start an injected conversion is ignored when JCIP=1.

Bits 12:5  Reserved, must be kept at reset value.

Bit 4  **AWDF**: Analog watchdog  
0: No Analog watchdog event occurred  
1: The analog watchdog block detected voltage which crosses the value programmed in the DFSDM_FLTxAWLTR or DFSDM_FLTxAWHTR registers.  
This bit is set by hardware. It is cleared by software by clearing all source flag bits AWHTF[7:0] and AWLTF[7:0] in DFSDM_FLTxAWSR register (by writing ‘1’ into the clear bits in DFSDM_FLTxAWCFR register).

Bit 3  **ROVRF**: Regular conversion overrun flag  
0: No regular conversion overrun has occurred  
1: A regular conversion overrun has occurred, which means that a regular conversion finished while REOCF was already ‘1’. RDATAR is not affected by overruns  
This bit is set by hardware. It can be cleared by software using the CLRROVRF bit in the DFSDM_FLTxCIR register.

Bit 2  **JOVRF**: Injected conversion overrun flag  
0: No injected conversion overrun has occurred  
1: An injected conversion overrun has occurred, which means that an injected conversion finished while JEOCF was already ‘1’. JDATAR is not affected by overruns  
This bit is set by hardware. It can be cleared by software using the CLRJOVRF bit in the DFSDM_FLTxCIR register.

Bit 1  **REOCF**: End of regular conversion flag  
0: No regular conversion has completed  
1: A regular conversion has completed and its data may be read  
This bit is set by hardware. It is cleared when the software or DMA reads DFSDM_FLTxDIFDAT.

Bit 0  **JEOCF**: End of injected conversion flag  
0: No injected conversion has completed  
1: An injected conversion has completed and its data may be read  
This bit is set by hardware. It is cleared when the software or DMA reads DFSDM_FLTxDIFDAT.

**Note:**  For each of the flag bits, an interrupt can be enabled by setting the corresponding bit in DFSDM_FLTxCIR2. If an interrupt is called, the flag must be cleared before exiting the interrupt service routine.

All the bits of DFSDM_FLTxhr are automatically reset when DFEN=0.
33.8.4 **DFSDM filter x interrupt flag clear register (DFSDM_FLTxICR)**

Address offset: 0x10C + 0x80 * x, (x = 0 to 7)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:24</th>
<th>CLRSCDF[7:0]</th>
<th>Clear the short-circuit detector flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CLRSCDF[y]=0: Writing ‘0’ has no effect</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLRSCDF[y]=1: Writing ‘1’ to position y clears the corresponding SCDF[y] bit in the DFSDM_FLTxISR register</td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td>CLRSCDF[7:0] is present only in DFSDM_FLT0ICR register (filter x=0)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 23:16</th>
<th>CLRCKABF[7:0]</th>
<th>Clear the clock absence flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CLRCKABF[y]=0: Writing ‘0’ has no effect</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLRCKABF[y]=1: Writing ‘1’ to position y clears the corresponding CKABF[y] bit in the DFSDM_FLTxISR register. When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y].</td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td>CLRCKABF[7:0] is present only in DFSDM_FLT0ICR register (filter x=0)</td>
<td></td>
</tr>
</tbody>
</table>

| Bits 15:4  | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>CLRROVRF: Clear the regular conversion overrun flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Writing ‘0’ has no effect</td>
<td></td>
</tr>
<tr>
<td>1: Writing ‘1’ clears the ROVRF bit in the DFSDM_FLTxISR register</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>CLRJOVRF: Clear the injected conversion overrun flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Writing ‘0’ has no effect</td>
<td></td>
</tr>
<tr>
<td>1: Writing ‘1’ clears the JOVRF bit in the DFSDM_FLTxISR register</td>
<td></td>
</tr>
</tbody>
</table>

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

*Note:* The bits of DFSDM_FLTxICR are always read as ‘0’.
33.8.5  **DFSDM filter x injected channel group selection register (DFSDM_FLTxJCHGR)**

Address offset: 0x110 + 0x80 * x, (x = 0 to 7)

Reset value: 0x0000 0001

| Bits 31:8 | Reserved, must be kept at reset value. |
| Bits 7:0 | **JCHG[7:0]**: Injected channel group selection |
| JCHG[y]=0: channel y is not part of the injected group |
| JCHG[y]=1: channel y is part of the injected group |
| If JSCAN=1, each of the selected channels is converted, one after another. The lowest channel (channel 0, if selected) is converted first and the sequence ends at the highest selected channel. |
| If JSCAN=0, then only one channel is converted from the selected channels, and the channel selection is moved to the next channel. Writing JCHG, if JSCAN=0, resets the channel selection to the lowest selected channel. |
| At least one channel must always be selected for the injected group. Writes causing all JCHG bits to be zero are ignored. |

33.8.6  **DFSDM filter x control register (DFSDM_FLTxFCR)**

Address offset: 0x114 + 0x80 * x, (x = 0 to 7)

Reset value: 0x0000 0000

| Bits 31:8 | Reserved, must be kept at reset value. |
| Bits 7:0 | **FORD[2:0]** |
| Bits 15:8 | **FOSR[9:0]** |
| Bits 15:8 | **IOSR[7:0]** |
33.8.7 DFSDM filter x data register for injected group (DFSDM_FLTxJDATAR)

Address offset: 0x118 + 0x80 * x, (x = 0 to 7)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDATA[23:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>JDATA[7:0]</td>
<td>JDATA[2:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**33.8.8 DFSDM filter x data register for the regular channel (DFSDM_FLTxRDATAR)**

Address offset: \(0x11C + 0x80 \times x\), \((x = 0\) to \(7)\)

Reset value: \(0x0000\) \(0000\)

Bits 31:8 **JDATA[23:0]**: Injected group conversion data

When each conversion of a channel in the injected group finishes, its resulting data is stored in this field. The data is valid when JEOCF=1. Reading this register clears the corresponding JEOCF.

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

Bits 2:0 **JDATACH[2:0]**: Injected channel most recently converted

When each conversion of a channel in the injected group finishes, JDATACH[2:0] is updated to indicate which channel was converted. Thus, JDATA[23:0] holds the data that corresponds to the channel indicated by JDATACH[2:0].

**Note:** DMA may be used to read the data from this register. Half-word accesses may be used to read only the MSBs of conversion data.

Reading this register also clears JEOCF in DFSDM_FLTxISR. Thus, the firmware must not read this register if DMA is activated to read data from this register.

**Bits 31:8** **RDATA[23:0]**: Regular channel conversion data

When each regular conversion finishes, its data is stored in this register. The data is valid when REOCF=1. Reading this register clears the corresponding REOCF.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **RPEND**: Regular channel pending data

Regular data in RDATA[23:0] was delayed due to an injected channel trigger during the conversion

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **RDATACH[2:0]**: Regular channel most recently converted

When each regular conversion finishes, RDATACH[2:0] is updated to indicate which channel was converted (because regular channel selection RCH[2:0] in DFSDM_FLTxCR1 register can be updated during regular conversion). Thus RDATA[23:0] holds the data that corresponds to the channel indicated by RDATACH[2:0].

**Note:** Half-word accesses may be used to read only the MSBs of conversion data.

Reading this register also clears REOCF in DFSDM_FLTxISR.
### 33.8.9 DFSDM filter x analog watchdog high threshold register (DFSDM_FLTxAWHTR)

Address offset: $0x120 + 0x80 \times x$, ($x = 0$ to 7)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Description</th>
<th>Reset Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-8</td>
<td>AWHT[23:8]: Analog watchdog high threshold</td>
<td>0x0000 0000</td>
<td>These bits are written by software to define the high threshold for the analog watchdog. In case channel transceivers monitor (AWFSEL=1), the higher 16 bits (AWHT[23:8]) define the 16-bit threshold as compared with the analog watchdog filter output (because data coming from the analog watchdog filter are up to a 16-bit resolution). Bits AWHT[7:0] are not taken into comparison in this case.</td>
</tr>
<tr>
<td>7-4</td>
<td>Reserved</td>
<td>0x0000 0000</td>
<td></td>
</tr>
<tr>
<td>3-0</td>
<td>BKAWH[3:0]: Break signal assignment to analog watchdog high threshold event</td>
<td>0x0000 0000</td>
<td>BKAWH[i] = 0: Break i signal is not assigned to an analog watchdog high threshold event. BKAWH[i] = 1: Break i signal is assigned to an analog watchdog high threshold event</td>
</tr>
</tbody>
</table>

### 33.8.10 DFSDM filter x analog watchdog low threshold register (DFSDM_FLTxAWLTR)

Address offset: $0x124 + 0x80 \times x$, ($x = 0$ to 7)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Description</th>
<th>Reset Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-8</td>
<td>AWLT[23:8]: Analog watchdog low threshold</td>
<td>0x0000 0000</td>
<td></td>
</tr>
<tr>
<td>7-4</td>
<td>Reserved</td>
<td>0x0000 0000</td>
<td></td>
</tr>
<tr>
<td>3-0</td>
<td>BKAWL[3:0]: Break signal assignment to analog watchdog low threshold event</td>
<td>0x0000 0000</td>
<td></td>
</tr>
</tbody>
</table>
33.8.11 DFSDM filter x analog watchdog status register (DFSDM_FLTxAWSR)

Address offset: 0x128 + 0x80 * x, (x = 0 to 7)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits 15:8</th>
<th>AWHTF[7:0]: Analog watchdog high threshold flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AWHTF[y]=1 indicates a high threshold error on channel y. It is set by hardware. It can be cleared by software using the corresponding CLRAWHTF[y] bit in the DFSDM_FLTxAWCFR register.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 7:0</th>
<th>AWLTF[7:0]: Analog watchdog low threshold flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AWLTF[y]=1 indicates a low threshold error on channel y. It is set by hardware. It can be cleared by software using the corresponding CLRAWLTF[y] bit in the DFSDM_FLTxAWCFR register.</td>
</tr>
</tbody>
</table>

Note: All the bits of DFSDM_FLTxAWSR are automatically reset when DFEN=0.
33.8.12 **DFSDM filter x analog watchdog clear flag register**  
(DFSDM_FLTxAWCFR)

Address offset: $0x12C + 0x80 \times x$, ($x = 0$ to $7$)

Reset value: $0x0000\ 0000$

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

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLRAWHTF[7:0]</td>
<td>CLRAWLTF[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **CLRAWHTF[7:0]**: Clear the analog watchdog high threshold flag

- $CLRAWHTF[y]=0$: Writing ‘0’ has no effect
- $CLRAWHTF[y]=1$: Writing ‘1’ to position $y$ clears the corresponding $AWHTF[y]$ bit in the DFSDM_FLTxAWSR register

Bits 7:0 **CLRAWLTF[7:0]**: Clear the analog watchdog low threshold flag

- $CLRAWLTF[y]=0$: Writing ‘0’ has no effect
- $CLRAWLTF[y]=1$: Writing ‘1’ to position $y$ clears the corresponding $AWLTF[y]$ bit in the DFSDM_FLTxAWSR register

33.8.13 **DFSDM filter x extremes detector maximum register**  
(DFSDM_FLTxEXMAX)

Address offset: $0x130 + 0x80 \times x$, ($x = 0$ to $7$)

Reset value: $0x8000\ 0000$

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

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXMAX[23:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| EXMAX[7:0] | EXMAXCH[2:0] |
| rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r |
| 15  | 14  | 13  | 12  | 11  | 10  |  9  |  8  |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |

Bits 31:8 **EXMAX[23:0]**: Extremes detector maximum value

These bits are set by hardware and indicate the highest value converted by DFSDM_FLTx. 
EXMAX[23:0] bits are reset to value (0x8000000) by reading of this register.

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

Bits 2:0 **EXMAXCH[2:0]**: Extremes detector maximum data channel.

These bits contains information about the channel on which the data is stored into EXMAX[23:0].

Bits are cleared by reading of this register.
33.8.14 **DFSDM filter x extremes detector minimum register (DFSDM_FLTxEXMIN)**

Address offset: 0x134 + 0x80 * x, (x = 0 to 7)

Reset value: 0x7FFF FF00

<table>
<thead>
<tr>
<th>EXMIN[23:8]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>EXMINCH[2:0]</td>
</tr>
<tr>
<td>rs_r rs_r rs_r rs_r rs_r rs_r rs_r rs_r rs_r rs_r rs_r rs_r rs_r rs_r rs_r rs_r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 **EXMIN[23:0]**: Extremes detector minimum value

These bits are set by hardware and indicate the lowest value converted by DFSDM_FLTx.

EXMIN[23:0] bits are reset to value (0x7FFFFFFF) by reading of this register.

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

Bits 2:0 **EXMINCH[2:0]**: Extremes detector minimum data channel

These bits contain information about the channel on which the data is stored into EXMIN[23:0]. Bits are cleared by reading of this register.

33.8.15 **DFSDM filter x conversion timer register (DFSDM_FLTxCNVTIMR)**

Address offset: 0x138 + 0x80 * x, (x = 0 to 7)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>CNVCNT[27:12]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r</td>
<td></td>
</tr>
</tbody>
</table>

Bits are cleared by reading of this register.
33.8.16 DFSDM register map

The following table summarizes the DFSDM registers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 31:4</th>
<th>Bits 3:0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>DFSDM_CH0CFGR1</td>
<td>CKOUTDIV[7:0]</td>
<td>Reserved</td>
<td>CNVCNT[27:0]: 28-bit timer counting conversion time t = CNVCNT[27:0] / fDFSDMCLK. The timer has an input clock from DFSDM clock (system clock fDFSDMCLK). Conversion time measurement is started on each conversion start and stopped when conversion finishes (interval between first and last serial sample). Only in case of filter bypass (FOSR[9:0] = 0) is the conversion time measurement stopped and CNVCNT[27:0] = 0. The counted time is: if FAST=0 (or first conversion in continuous mode if FAST=1): t = [FOSR * (IOSR-1 + FORD) + FORD] / fCKIN..... for Sincx filters t = [FOSR * (IOSR-1 + 4) + 2] / fCKIN..... for FastSinc filter if FAST=1 in continuous mode (except first conversion): t = [FOSR * IOSR] / fCKIN in case if FOSR = FOSR[9:0]+1 = 1 (filter bypassed, active only integrator): CNVCNT = 0 (counting is stopped, conversion time: t = IOSR / fCKIN) where: fCKIN is the channel input clock frequency (on given channel CKINy pin) or input data rate in case of parallel data input (from internal ADC or from CPU/DMA write) Note: When conversion is interrupted (e.g. by disable/enable selected channel) the timer counts also this interruption time.</td>
</tr>
<tr>
<td>0x04</td>
<td>DFSDM_CH0CFGR2</td>
<td>OFFSET[23:0]</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>DFSDM_CH0AWSCDR</td>
<td>AWFORD[1:0]</td>
<td>BKSCD[3:0]</td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>DFSDM_CH0WDATR</td>
<td>WDATA[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>DFSDM_CH0DATINR</td>
<td>INDAT1[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>DFSDM_CH0DLYR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x18 - 0x1C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 246. DFSDM register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x20</td>
<td>DFSDM CH1CFGR1</td>
<td>DFSDM CH1CFGR1</td>
<td>DFSDM CH1CFGR1</td>
<td>DFSDM CH1CFGR1</td>
<td>DFSDM CH1CFGR1</td>
<td>DFSDM CH1CFGR1</td>
<td>DFSDM CH1CFGR1</td>
<td>DFSDM CH1CFGR1</td>
<td>DFSDM CH1CFGR1</td>
<td>DFSDM CH1CFGR1</td>
</tr>
<tr>
<td>0x24</td>
<td>DFSDM CH1CFGR2</td>
<td>DFSDM CH1CFGR2</td>
<td>DFSDM CH1CFGR2</td>
<td>DFSDM CH1CFGR2</td>
<td>DFSDM CH1CFGR2</td>
<td>DFSDM CH1CFGR2</td>
<td>DFSDM CH1CFGR2</td>
<td>DFSDM CH1CFGR2</td>
<td>DFSDM CH1CFGR2</td>
<td>DFSDM CH1CFGR2</td>
</tr>
<tr>
<td>0x28</td>
<td>DFSDM CH1AWSCDR</td>
<td>DFSDM CH1AWSCDR</td>
<td>DFSDM CH1AWSCDR</td>
<td>DFSDM CH1AWSCDR</td>
<td>DFSDM CH1AWSCDR</td>
<td>DFSDM CH1AWSCDR</td>
<td>DFSDM CH1AWSCDR</td>
<td>DFSDM CH1AWSCDR</td>
<td>DFSDM CH1AWSCDR</td>
<td>DFSDM CH1AWSCDR</td>
</tr>
<tr>
<td>0x2C</td>
<td>DFSDM CH1WDATR</td>
<td>DFSDM CH1WDATR</td>
<td>DFSDM CH1WDATR</td>
<td>DFSDM CH1WDATR</td>
<td>DFSDM CH1WDATR</td>
<td>DFSDM CH1WDATR</td>
<td>DFSDM CH1WDATR</td>
<td>DFSDM CH1WDATR</td>
<td>DFSDM CH1WDATR</td>
<td>DFSDM CH1WDATR</td>
</tr>
<tr>
<td>0x30</td>
<td>DFSDM CH1DATINR</td>
<td>DFSDM CH1DATINR</td>
<td>DFSDM CH1DATINR</td>
<td>DFSDM CH1DATINR</td>
<td>DFSDM CH1DATINR</td>
<td>DFSDM CH1DATINR</td>
<td>DFSDM CH1DATINR</td>
<td>DFSDM CH1DATINR</td>
<td>DFSDM CH1DATINR</td>
<td>DFSDM CH1DATINR</td>
</tr>
<tr>
<td>0x34</td>
<td>DFSDM CH1DLYR</td>
<td>DFSDM CH1DLYR</td>
<td>DFSDM CH1DLYR</td>
<td>DFSDM CH1DLYR</td>
<td>DFSDM CH1DLYR</td>
<td>DFSDM CH1DLYR</td>
<td>DFSDM CH1DLYR</td>
<td>DFSDM CH1DLYR</td>
<td>DFSDM CH1DLYR</td>
<td>DFSDM CH1DLYR</td>
</tr>
<tr>
<td>0x38 - 0x3C</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x40</td>
<td>DFSDM CH2CFGR1</td>
<td>DFSDM CH2CFGR1</td>
<td>DFSDM CH2CFGR1</td>
<td>DFSDM CH2CFGR1</td>
<td>DFSDM CH2CFGR1</td>
<td>DFSDM CH2CFGR1</td>
<td>DFSDM CH2CFGR1</td>
<td>DFSDM CH2CFGR1</td>
<td>DFSDM CH2CFGR1</td>
<td>DFSDM CH2CFGR1</td>
</tr>
<tr>
<td>0x44</td>
<td>DFSDM CH2CFGR2</td>
<td>DFSDM CH2CFGR2</td>
<td>DFSDM CH2CFGR2</td>
<td>DFSDM CH2CFGR2</td>
<td>DFSDM CH2CFGR2</td>
<td>DFSDM CH2CFGR2</td>
<td>DFSDM CH2CFGR2</td>
<td>DFSDM CH2CFGR2</td>
<td>DFSDM CH2CFGR2</td>
<td>DFSDM CH2CFGR2</td>
</tr>
<tr>
<td>0x48</td>
<td>DFSDM CH2AWSCDR</td>
<td>DFSDM CH2AWSCDR</td>
<td>DFSDM CH2AWSCDR</td>
<td>DFSDM CH2AWSCDR</td>
<td>DFSDM CH2AWSCDR</td>
<td>DFSDM CH2AWSCDR</td>
<td>DFSDM CH2AWSCDR</td>
<td>DFSDM CH2AWSCDR</td>
<td>DFSDM CH2AWSCDR</td>
<td>DFSDM CH2AWSCDR</td>
</tr>
<tr>
<td>0x4C</td>
<td>DFSDM CH2WDATR</td>
<td>DFSDM CH2WDATR</td>
<td>DFSDM CH2WDATR</td>
<td>DFSDM CH2WDATR</td>
<td>DFSDM CH2WDATR</td>
<td>DFSDM CH2WDATR</td>
<td>DFSDM CH2WDATR</td>
<td>DFSDM CH2WDATR</td>
<td>DFSDM CH2WDATR</td>
<td>DFSDM CH2WDATR</td>
</tr>
<tr>
<td>0x50</td>
<td>DFSDM CH2DATINR</td>
<td>DFSDM CH2DATINR</td>
<td>DFSDM CH2DATINR</td>
<td>DFSDM CH2DATINR</td>
<td>DFSDM CH2DATINR</td>
<td>DFSDM CH2DATINR</td>
<td>DFSDM CH2DATINR</td>
<td>DFSDM CH2DATINR</td>
<td>DFSDM CH2DATINR</td>
<td>DFSDM CH2DATINR</td>
</tr>
<tr>
<td>0x54</td>
<td>DFSDM CH2DLYR</td>
<td>DFSDM CH2DLYR</td>
<td>DFSDM CH2DLYR</td>
<td>DFSDM CH2DLYR</td>
<td>DFSDM CH2DLYR</td>
<td>DFSDM CH2DLYR</td>
<td>DFSDM CH2DLYR</td>
<td>DFSDM CH2DLYR</td>
<td>DFSDM CH2DLYR</td>
<td>DFSDM CH2DLYR</td>
</tr>
<tr>
<td>0x58 - 0x5C</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Register Map and Reset Values:**
- Offset: Register offset in memory.
- Name: Name of the register.
- Description: Description of the register and its functions.
- Reset Value: The default or reset value for the register.

The table provides a detailed view of the DFSDM register map, including the offset for each register and the reset value for each register field. The registers are labeled with their names and their locations within the memory map are indicated. Each register includes a description of its function and the reset value for each bit position within the register. The table is structured to allow for easy reference and understanding of the register settings for the DFSDM module.
## Table 246. DFSDM register map and reset values (continued)

| Offset | Register name       | Offset Register | name                      | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------------|-----------------|---------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x60   | DFSDM_CH3CFGR1     |                 | register value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x64   | DFSDM_CH3CFGR2     | OFFSET[23:0]    | DTRBS[4:0]                | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x68   | DFSDM_CH3AWSCDR    |                 |                           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x6C   | DFSDM_CH3WDATR     |                 | WDATA[15:0]               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x70   | DFSDM_CH3DATINR    | INDAT[15:0]     | INDATO[15:0]              | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x74   | DFSDM_CH3DLYR      |                 |                           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x78 - | Reserved            |                 |                           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x80   | DFSDM_CH4CFGR1     |                 |                           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x84   | DFSDM_CH4CFGR2     | OFFSET[23:0]    | DTRBS[4:0]                | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x88   | DFSDM_CH4AWSCDR    |                 |                           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x8C   | DFSDM_CH4WDATR     |                 | WDATA[15:0]               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x90   | DFSDM_CH4DATINR    | INDAT[15:0]     | INDATO[15:0]              | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x94   | DFSDM_CH4DLYR      |                 |                           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x98 - | Reserved            |                 |                           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Reset value 0000 0000 0000
| Offset | Register name       | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|--------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0xA0   | DFSDM_CH5CFGR1     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xA4   | DFSDM_CH5CFGR2     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | reset value |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xA8   | DFSDM_CH5AWSCDR    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | reset value |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xAC   | DFSDM_CH5WDATR     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | reset value |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xB0   | DFSDM_CH5DATInR    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | reset value |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xB4   | DFSDM_CH5DLYR      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | reset value |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xB8 - | Reserved           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xBC   | Reserved           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xC0   | DFSDM_CH6CFGR1     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | reset value |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xC4   | DFSDM_CH6CFGR2     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | reset value |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xC8   | DFSDM_CH6AWSCDR    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | reset value |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xCC   | DFSDM_CH6WDATR     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | reset value |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xD0   | DFSDM_CH6DATInR    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | reset value |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xD4   | DFSDM_CH6DLYR      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | reset value |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xDC   | Reserved           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Table 246: DFSDM register map and reset values (continued)
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE0</td>
<td>DFSDM_CH7CFGR1</td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0xE4</td>
<td>DFSDM_CH7CFGR2</td>
<td>OFFSET[23:0] DTRBS[4:0]</td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0xE8</td>
<td>DFSDM_CH7AWSCDR</td>
<td>AWFOSR[4:0] BKSCD[3:0] SCD[7:0]</td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0xEC</td>
<td>DFSDM_CH7WDATR</td>
<td>WDATA[15:0]</td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0xF0</td>
<td>DFSDM_CH7DATINR</td>
<td>INDAT1[15:0] INDAT0[15:0]</td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0xF4</td>
<td>DFSDM_CH7DLYR</td>
<td>PLSSKP[5:0]</td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x100</td>
<td>Reserved</td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x104</td>
<td>DFSDM_FLT0CR2</td>
<td>AWDCH[7:0] EXCH[7:0]</td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x108</td>
<td>DFSDM_FLT0ISR</td>
<td>SCDF[7:0] CKABF[7:0] RCIP JCP</td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x10C</td>
<td>DFSDM_FLT0ICR</td>
<td>CLRSCDF[7:0] CLRCKABF[7:0]</td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x110</td>
<td>DFSDM_FLT0JCHGR</td>
<td>JCHG[7:0]</td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x114</td>
<td>DFSDM_FLT0FCR</td>
<td>FOSR[9:0] IOSR[7:0]</td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
</tbody>
</table>
Table 246. DFSDM register map and reset values (continued)

| Offset | Register name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x118  | DFSDM_FLT0JDATAR |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x11C  | DFSDM_FLT0RDATAR |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x120  | DFSDM_FLT0AWHTR |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x124  | DFSDM_FLT0AWLTR |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x128  | DFSDM_FLT0AWSR |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x12C  | DFSDM_FLT0AWCFR |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x130  | DFSDM_FLT0EXMAX |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x134  | DFSDM_FLT0EXMIN |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | reset value    | 0   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x138  | DFSDM_FLT0CNVTIMR |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x13C - 0x17C | Reserved |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x180  | DFSDM_FLT1CR1  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x184  | DFSDM_FLT1CR2  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x188  | DFSDM_FLT1ISR  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
### Table 246. DFSDM register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Function</th>
<th>Offset</th>
<th>Register name</th>
<th>Function</th>
<th>Offset</th>
<th>Register name</th>
<th>Function</th>
<th>Offset</th>
<th>Register name</th>
<th>Function</th>
<th>Offset</th>
<th>Register name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x18C</td>
<td>DFSDM_FLT1ICR</td>
<td>reset value</td>
<td>0x190</td>
<td>DFSDM_FLT1JCHGR</td>
<td>JCHG[7:0]</td>
<td>0x194</td>
<td>DFSDM_FLT1IFCR</td>
<td>reset value</td>
<td>0x0000000000000000000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x198</td>
<td>DFSDM_FLT1JDATAR</td>
<td>JDATA[23:0]</td>
<td>0x19C</td>
<td>DFSDM_FLT1RDATAR</td>
<td>RDATA[23:0]</td>
<td>0x1A0</td>
<td>DFSDM_FLT1AWHTR</td>
<td>AWHT[23:0]</td>
<td>0x1A4</td>
<td>DFSDM_FLT1AWLTR</td>
<td>AWLT[23:0]</td>
<td>0x1A8</td>
<td>DFSDM_FLT1AWSR</td>
<td>AWHTF[7:0]</td>
</tr>
<tr>
<td>0x19C</td>
<td>DFSDM_FLT1JDATAR</td>
<td>JDATA[23:0]</td>
<td>0x198</td>
<td>DFSDM_FLT1JDATAR</td>
<td>JDATA[23:0]</td>
<td>0x19A</td>
<td>DFSDM_FLT1JCHGR</td>
<td>JCHG[7:0]</td>
<td>0x1AC</td>
<td>DFSDM_FLT1AWCTR</td>
<td>CLRAWHTF[7:0]</td>
<td>0x1A0</td>
<td>DFSDM_FLT1AWHTR</td>
<td>AWHT[23:0]</td>
</tr>
<tr>
<td>0x1A0</td>
<td>DFSDM_FLT1AWHTR</td>
<td>AWHT[23:0]</td>
<td>0x1A0</td>
<td>DFSDM_FLT1AWHTR</td>
<td>AWHT[23:0]</td>
<td>0x1A4</td>
<td>DFSDM_FLT1AWLTR</td>
<td>AWLT[23:0]</td>
<td>0x1A8</td>
<td>DFSDM_FLT1AWCTR</td>
<td>CLRAWHTF[7:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1A4</td>
<td>DFSDM_FLT1AWLTR</td>
<td>AWLT[23:0]</td>
<td>0x1A4</td>
<td>DFSDM_FLT1AWLTR</td>
<td>AWLT[23:0]</td>
<td>0x1A8</td>
<td>DFSDM_FLT1AWCTR</td>
<td>CLRAWHTF[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1A8</td>
<td>DFSDM_FLT1AWCTR</td>
<td>CLRAWHTF[7:0]</td>
<td>0x1AC</td>
<td>DFSDM_FLT1AWCFR</td>
<td>CLRAWHTF[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1B0</td>
<td>DFSDM_FLT1EXMAX</td>
<td>EXMAX[23:0]</td>
<td>0x1B0</td>
<td>DFSDM_FLT1EXMAX</td>
<td>EXMAX[23:0]</td>
<td>0x1B4</td>
<td>DFSDM_FLT1EXMIN</td>
<td>EXMIN[23:0]</td>
<td>0x1B6</td>
<td>DFSDM_FLT1CNPCTMR</td>
<td>CNVCT[27:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1B4</td>
<td>DFSDM_FLT1EXMIN</td>
<td>EXMIN[23:0]</td>
<td>0x1B4</td>
<td>DFSDM_FLT1EXMIN</td>
<td>EXMIN[23:0]</td>
<td>0x1B6</td>
<td>DFSDM_FLT1CNPCTMR</td>
<td>CNVCT[27:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1B6</td>
<td>DFSDM_FLT1CNPCTMR</td>
<td>CNVCT[27:0]</td>
<td>0x1B6</td>
<td>DFSDM_FLT1CNPCTMR</td>
<td>CNVCT[27:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The reset values are provided for each register and are shown in hexadecimal format.

---

**Table 246. DFSDM register map and reset values (continued)**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Function</th>
<th>Offset</th>
<th>Register name</th>
<th>Function</th>
<th>Offset</th>
<th>Register name</th>
<th>Function</th>
<th>Offset</th>
<th>Register name</th>
<th>Function</th>
<th>Offset</th>
<th>Register name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x18C</td>
<td>DFSDM_FLT1ICR</td>
<td>reset value</td>
<td>0x190</td>
<td>DFSDM_FLT1JCHGR</td>
<td>JCHG[7:0]</td>
<td>0x194</td>
<td>DFSDM_FLT1IFCR</td>
<td>reset value</td>
<td>0x0000000000000000000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x198</td>
<td>DFSDM_FLT1JDATAR</td>
<td>JDATA[23:0]</td>
<td>0x19C</td>
<td>DFSDM_FLT1RDATAR</td>
<td>RDATA[23:0]</td>
<td>0x1A0</td>
<td>DFSDM_FLT1AWHTR</td>
<td>AWHT[23:0]</td>
<td>0x1A4</td>
<td>DFSDM_FLT1AWLTR</td>
<td>AWLT[23:0]</td>
<td>0x1A8</td>
<td>DFSDM_FLT1AWSR</td>
<td>AWHTF[7:0]</td>
</tr>
<tr>
<td>0x1A0</td>
<td>DFSDM_FLT1AWHTR</td>
<td>AWHT[23:0]</td>
<td>0x1A0</td>
<td>DFSDM_FLT1AWHTR</td>
<td>AWHT[23:0]</td>
<td>0x1A4</td>
<td>DFSDM_FLT1AWLTR</td>
<td>AWLT[23:0]</td>
<td>0x1A8</td>
<td>DFSDM_FLT1AWCTR</td>
<td>CLRAWHTF[7:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1A4</td>
<td>DFSDM_FLT1AWLTR</td>
<td>AWLT[23:0]</td>
<td>0x1A4</td>
<td>DFSDM_FLT1AWLTR</td>
<td>AWLT[23:0]</td>
<td>0x1A8</td>
<td>DFSDM_FLT1AWCTR</td>
<td>CLRAWHTF[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1A8</td>
<td>DFSDM_FLT1AWCTR</td>
<td>CLRAWHTF[7:0]</td>
<td>0x1AC</td>
<td>DFSDM_FLT1AWCFR</td>
<td>CLRAWHTF[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1B0</td>
<td>DFSDM_FLT1EXMAX</td>
<td>EXMAX[23:0]</td>
<td>0x1B0</td>
<td>DFSDM_FLT1EXMAX</td>
<td>EXMAX[23:0]</td>
<td>0x1B4</td>
<td>DFSDM_FLT1EXMIN</td>
<td>EXMIN[23:0]</td>
<td>0x1B6</td>
<td>DFSDM_FLT1CNPCTMR</td>
<td>CNVCT[27:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1B4</td>
<td>DFSDM_FLT1EXMIN</td>
<td>EXMIN[23:0]</td>
<td>0x1B4</td>
<td>DFSDM_FLT1EXMIN</td>
<td>EXMIN[23:0]</td>
<td>0x1B6</td>
<td>DFSDM_FLT1CNPCTMR</td>
<td>CNVCT[27:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1B6</td>
<td>DFSDM_FLT1CNPCTMR</td>
<td>CNVCT[27:0]</td>
<td>0x1B6</td>
<td>DFSDM_FLT1CNPCTMR</td>
<td>CNVCT[27:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The reset values are provided for each register and are shown in hexadecimal format.
| Offset  | Register name          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x200   | DFSDM_FLT2CR1          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x204   | DFSDM_FLT2CR2          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x208   | DFSDM_FLT2ISR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x20C   | DFSDM_FLT2ICR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x210   | DFSDM_FLT2JCHGR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x214   | DFSDM_FLT2FCR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x218   | DFSDM_FLT2JDATAR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x21C   | DFSDM_FLT2RDATAR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x220   | DFSDM_FLT2AWHTR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x224   | DFSDM_FLT2AWLTR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x228   | DFSDM_FLT2AWSR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x22C   | DFSDM_FLT2AWCFR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Table 246. DFSDM register map and reset values (continued)
### Table 246. DFSDM register map and reset values (continued)

| Offset   | Register name             | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|---------------------------|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x230    | DFSDM_FLT2EXMAX           |     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |                           | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x234    | DFSDM_FLT2EXMIN           |     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |                           | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x238    | DFSDM_FLT2CNVTIMR         |     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |                           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x23C-   | 0x27C Reserved            |     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x280    | DFSDM_FLT3CR1             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |                           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x284    | DFSDM_FLT3CR2             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |                           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x288    | DFSDM_FLT3ISR             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |                           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x290    | DFSDM_FLT3DATAR           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |                           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x294    | DFSDM_FLT3FCR             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |                           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x298    | DFSDM_FLT3JCHGR           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |                           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

**Note:** The table above shows the register map and reset values for DFSDM, a digital filter for sigma delta modulators. The reset values are provided in hexadecimal format for each register offset.
Table 246. DFSDM register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Name</th>
<th>Offset</th>
<th>Register name</th>
<th>Name</th>
<th>Offset</th>
<th>Register name</th>
<th>Name</th>
<th>Offset</th>
<th>Register name</th>
<th>Name</th>
<th>Offset</th>
<th>Register name</th>
<th>Name</th>
<th>Offset</th>
<th>Register name</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x29C</td>
<td>DFSDM_FLT3RDATAR</td>
<td>RDATA[23:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2A0</td>
<td>DFSDM_FLT3AWHTR</td>
<td>AWH[23:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2A4</td>
<td>DFSDM_FLT3AWLTR</td>
<td>AWL[23:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2A8</td>
<td>DFSDM_FLT3AWSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2AC</td>
<td>DFSDM_FLT3AWCFR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2B0</td>
<td>DFSDM_FLT3EXMAX</td>
<td>EXM[23:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2B4</td>
<td>DFSDM_FLT3EXMIN</td>
<td>EXM[23:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2BB</td>
<td>DFSDM_FLT3CNVTMR</td>
<td>CNVT[23:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2BC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2FC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x300</td>
<td>AWSEL[2:0]</td>
<td>RCH[2:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x304</td>
<td>AWSEL[2:0]</td>
<td>EXC[2:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x308</td>
<td>AWSEL[2:0]</td>
<td>EXC[2:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x30C</td>
<td>AWSEL[2:0]</td>
<td>EXC[2:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 246. DFSDM register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x310</td>
<td>DFSDM FLT4JCHR</td>
<td>JCHG[7:0]</td>
<td>0x314</td>
<td>DFSDM FLT4FCR</td>
<td>FORD[2:0]</td>
<td>0x318</td>
<td>DFSDM FLT4JDATAR</td>
<td>JDATA[23:0]</td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 1</td>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x318</td>
<td>DFSDM FLT4JCHR</td>
<td>JCHG[7:0]</td>
<td>0x31C</td>
<td>DFSDM FLT4RDATA</td>
<td>RDATA[23:0]</td>
<td>0x320</td>
<td>DFSDM FLT4AWLTR</td>
<td>AWLT[23:0]</td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x320</td>
<td>DFSDM FLT4AWHT</td>
<td>AWH[7:0]</td>
<td>0x324</td>
<td>DFSDM FLT4AWLTR</td>
<td>AWLT[23:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x328</td>
<td>DFSDM FLT4AWHTR</td>
<td>AWH[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x32C</td>
<td>DFSDM FLT4AWCTR</td>
<td>AWH[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x330</td>
<td>DFSDM FLT4EXMAX</td>
<td>EXMAX[23:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>1 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x334</td>
<td>DFSDM FLT4EXMIN</td>
<td>EXMIN[23:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x338</td>
<td>DFSDM FLT4CNVTIMR</td>
<td>CNVTIMR[27:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x33C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x380</td>
<td>DFSDM FLT5CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>0x384</td>
<td>DFSDMFLTSCR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x388</td>
<td>DFSDMFLTSSISR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x38C</td>
<td>DFSDMFLTSCIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x390</td>
<td>DFSDMFLTSHGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x394</td>
<td>DFSDMFLT5FCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x398</td>
<td>DFSDMFLT5JDATAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x39C</td>
<td>DFSDMFLT5RDATAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x3A0</td>
<td>DFSDMFLT5AWHTR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x3A4</td>
<td>DFSDMFLT5SAWLTR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x3A8</td>
<td>DFSDMFLT5AWSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x3AC</td>
<td>DFSDMFLT5AFCFR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x3B0</td>
<td>DFSDMFLT5EXMÄX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>0x3B4</td>
<td>DFSDM_FLT5EXMIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0x3B8</td>
<td>DFSDMFLT5CNVTIMR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x3BC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x400</td>
<td>DFSDM_FLT6CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x404</td>
<td>DFSDM_FLT6CR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x408</td>
<td>DFSDM_FLT6ISR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x40C</td>
<td>DFSDM_FLT6ICR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x410</td>
<td>DFSDM_FLT6JCHGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x414</td>
<td>DFSDM_FLT6JCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x418</td>
<td>DFSDM_FLT6JDATAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x41C</td>
<td>DFSDM_FLT6RDATAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x420</td>
<td>DFSDM_FLT6AWHT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 246. DFSDM register map and reset values (continued)
0x494

reset value

1212/2974

DFSDM_
FLT7FCR

0

0

0

0

0

0

0

0

0

0

0

0

RM0455 Rev 3

FOSR[9:0]

0
0
0

0
0
0
0
0
0

reset value

0

0

0

0

reset value

0

0

0

0

0

Res.
Res.

0

Res.

0

DFEN

0

JEOCIE

0

JEOCF

0

JEXTSEL[4:0]

EXMINCH[2:0]

Res.

Res.

Res.
Res.
EXMAXCH[2:0]

Res.

Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.

0

0

0
0
0
0
0

0

0

JCHG[7:0]

0

IOSR[7:0]

Res.

CNVCNT[27:0]
Res.

1

0

Res.

1

0

Res.

1

0

Res.

1

0

JSW START

1

JOVRIE

1

JOVRF

1

REOCF

1
Res.

Res.

0

Res.

1

CLR JOVRF

1

Res.

0

Res.

0

JSYNC

0

ROVRF

0

CLR ROVRF

0

Res.

0

JSCAN

0

AWDF

0

Res.

0

JDMAEN

0
0

Res.

EXMAX[23:0]
0

Res.

0

Res.

0

Res.

EXMIN[23:0]
Res.

0

0

Res.

0

Res.

CLRAWHTF[7:0]

Res.

0

Res.

0

0

Res.

0

Res.

0

Res.

0

Res.

0

Res.

0

Res.

0

0

Res.

EXCH[7:0]
Res.

0

Res.

0

Res.

0

Res.

0

0

REOCIE

0

Res.

0

Res.

Res.

AWHTF[7:0]

ROVRIE

0

0
0

0

Res.

0

Res.

Res.

Res.

Res.

Res.

Res.

AWLT[23:0]

AWDIE

0

0
0

0

Res.

0

Res.

AWDCH[7:0]
0

0

Res.

0

0

0

Res.

reset value
0

0

Res.

0

0

0

Res.

0

Res.

reset value
0

Res.

0

0

Res.

0

Res.

reset value
0

Res.

0

Res.

0

Res.

0

Res.

0

Res.

0

Res.

0

Res.

0

Res.

0

Res.

0

Res.

0

Res.

0

JEXTEN[1:0]

0

JCIP

0

RCIP

0

Res.

0

Res.

0

Res.

0

Res.

0

Res.

Res.

0

Res.

0

Res.

0

Res.

0

Res.

0

Res.

0

Res.

0

Res.

Res.

0

DFSDM_
FLT6AWSR

Res.

Res.

0

RSW START Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

reset value

Res.

Res.

0

Res.

1

Res.

Res.

RCONT

0
1

0

Res.

0

Res.

1

Res.

Res.

RSYNC

1
0

Res.

0

Res.

DFSDM_
FLT6CNVTIMR

Res.

Res.

Res.

1
0

Res.

0

Res.

DFSDM_
FLT6EXMIN

Res.

Res.

RDMAEN
0

Res.

0

Res.

0

Res.

Res.

Res.

DFSDM_
FLT6EXMAX

Res.

0

Res.

1

Res.

Res.

Res.
0

Res.

reset value

Res.

1

Res.

Res.

0

Res.
0

Res.

Res.

0

Res.
1

Res.

0

0

Res.

Res.

0

0

Res.
0

Res.

Res.

Res.

Res.

0

Res.
1

Res.

Res.

Res.

Res.

0

Res.

Res.

Res.

Res.

FAST

0

Res.
RCH[2:0]

Res.

1

Res.

Res.

Res.

Res.

Res.

DFSDM_
FLT6AWLTR

Res.

DFSDM_
FLT7JCHGR
0
0

Res.

DFSDM_
FLT7ICR
1

Res.

0x48C
0

Res.

DFSDM_
FLT7ISR
Res.

DFSDM_
FLT7CR2
Res.

reset value
1
0

Res.

0x490
DFSDM_
FLT7CR1

Res.

0x488
Reserved
0
0

Res.

0x484
0
1

Res.

0x480
0
0

Res.

0x43C 0x47C
reset value

AWFSEL

0x438

Res.

reset value

Res.

0x434
1

Res.

reset value

Res.

0x430

Res.

0x42C
DFSDM_
FLT6AWCFR

Res.

0x428

Res.

0x424

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

Register
name

Res.

Offset

FORD[2:0]

Digital filter for sigma delta modulators (DFSDM)
RM0455

Table 246. DFSDM register map and reset values (continued)

BKAWL[3:0]
0

AWLTF[7:0]
0

CLRAWLTF[7:0]
0

0

0

0
0

0
0

0
0

0

0

0

0

0
0

0
0
0
0
0

0

0

1

0

0

0


## Table 246. DFSDM register map and reset values (continued)

| Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset |Register to Section 2.3 on page 128 for the register boundary addresses.
34  Digital camera interface (DCMI)

34.1  Introduction

The digital camera is a synchronous parallel interface able to receive a high-speed data flow from an external 8-, 10-, 12- or 14-bit CMOS camera module. It supports different data formats: YCbCr4:2:2/RGB565 progressive video and compressed data (JPEG).

34.2  DCMI main features

- 8-, 10-, 12- or 14-bit parallel interface
- Embedded/external line and frame synchronization
- Continuous or snapshot mode
- Crop feature
- Supports the following data formats:
  - 8/10/12/14-bit progressive video: either monochrome or raw Bayer
  - YCbCr 4:2:2 progressive video
  - RGB 565 progressive video
  - Compressed data: JPEG

34.3  DCMI functional description

The digital camera interface is a synchronous parallel interface that can receive high-speed data flows. It consists of up to 14 data lines (DCMI_D[13:0]) and a pixel clock line (DCMI_PIXCLK). The pixel clock has a programmable polarity, so that data can be captured on either the rising or the falling edge of the pixel clock.

The data are packed into a 32-bit data register (DCMI_DR) and then transferred through a general-purpose DMA channel. The image buffer is managed by the DMA, not by the camera interface.

The data received from the camera can be organized in lines/frames (raw YUB/RGB/Bayer modes) or can be a sequence of JPEG images. To enable JPEG image reception, the JPEG bit (bit 3 of DCMI_CR register) must be set.

The data flow is synchronized either by hardware using the optional DCMI_HSYNC (horizontal synchronization) and DCMI_VSYNC (vertical synchronization) signals or by synchronization codes embedded in the data flow.
34.3.1 DCMI block diagram

*Figure 257* shows the DCMI block diagram.

*Figure 257. DCMI block diagram*

![DCMI block diagram](image)

34.3.2 DCMI pins and internal signals

The following table shows DCMI pins.

**Table 247. DCMI input/output pins**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>DCMI_D[7:0]</td>
<td>Inputs</td>
<td>DCMI data</td>
</tr>
<tr>
<td>10 bits</td>
<td>DCMI_D[9:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 bits</td>
<td>DCMI_D[11:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 bits</td>
<td>DCMI_D[13:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DCMI_PIXCLK</td>
<td>Input</td>
<td>Pixel clock</td>
</tr>
<tr>
<td></td>
<td>DCMI_HSYNC</td>
<td>Input</td>
<td>Horizontal synchronization / Data valid</td>
</tr>
<tr>
<td></td>
<td>DCMI_VSYNC</td>
<td>Input</td>
<td>Vertical synchronization</td>
</tr>
</tbody>
</table>

The following table shows DCMI internal signals.

**Table 248. DCMI internal input/output signals**

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcmi_dma</td>
<td>Output</td>
<td>DCMI DMA request</td>
</tr>
<tr>
<td>dcmi_it</td>
<td>Output</td>
<td>DCMI interrupt request</td>
</tr>
<tr>
<td>dcmi_hclk</td>
<td>Input</td>
<td>DCMI interface clock</td>
</tr>
</tbody>
</table>
34.3.3 DCMI clocks

The digital camera interface uses two clock domains, DCMI_PIXCLK and HCLK. The signals generated with DCMI_PIXCLK are sampled on the rising edge of HCLK once they are stable. An enable signal is generated in the HCLK domain, to indicate that data coming from the camera are stable and can be sampled. The maximum DCMI_PIXCLK period must be higher than 2.5 HCLK periods.

34.3.4 DCMI DMA interface

The DMA interface is active when the CAPTURE bit of the DCMI_CR register is set. A DMA request is generated each time the camera interface receives a complete 32-bit data block in its register.

34.3.5 DCMI physical interface

The interface is composed of 11/13/15/17 inputs. Only the Slave mode is supported.

The camera interface can capture 8-bit, 10-bit, 12-bit or 14-bit data depending on the EDM[1:0] bits of the DCMI_CR register. If less than 14 bits are used, the unused input pins must be connected to ground.

DCMI pins are shown in Table 247.

The data are synchronous with DCMI_PIXCLK and change on the rising/falling edge of the pixel clock depending on the polarity.

The DCMI_HSYNC signal indicates the start/end of a line.

The DCMI_VSYNC signal indicates the start/end of a frame.

![DCMI signal waveforms](image)

1. The capture edge of DCMI_PIXCLK is the falling edge, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

8-bit data

When EDM[1:0] = 00 in DCMI_CR the interface captures 8 LSBs at its input (DCMI_D[7:0]) and stores them as 8-bit data. The DCMI_D[13:8] inputs are ignored. In this case, to capture a 32-bit word, the camera interface takes four pixel clock cycles.

The first captured data byte is placed in the LSB position in the 32-bit word and the 4th captured data byte is placed in the MSB position in the 32-bit word. The table below gives an example of the positioning of captured data bytes in two 32-bit words.
When EDM[1:0] = 01 in DCMI_CR, the camera interface captures 10-bit data at its input DCMI_D[9:0] and stores them as the 10 least significant bits of a 16-bit word. The remaining most significant bits of the DCMI_DR register (bits 11 to 15) are cleared to zero. So, in this case, a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

### Table 249. Positioning of captured data bytes in 32-bit words (8-bit width)

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:24</th>
<th>23:16</th>
<th>15:8</th>
<th>7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D_{n+3}[7:0]</td>
<td>D_{n+2}[7:0]</td>
<td>D_{n+1}[7:0]</td>
<td>D_{n}[7:0]</td>
</tr>
<tr>
<td>4</td>
<td>D_{n+7}[7:0]</td>
<td>D_{n+6}[7:0]</td>
<td>D_{n+5}[7:0]</td>
<td>D_{n+4}[7:0]</td>
</tr>
</tbody>
</table>

### 10-bit data

When EDM[1:0] = 10 in DCMI_CR, the camera interface captures the 12-bit data at its input DCMI_D[11:0] and stores them as the 12 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

### Table 250. Positioning of captured data bytes in 32-bit words (10-bit width)

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:26</th>
<th>25:16</th>
<th>15:10</th>
<th>9:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>D_{n+1}[9:0]</td>
<td>0</td>
<td>D_{n}[9:0]</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>D_{n+3}[9:0]</td>
<td>0</td>
<td>D_{n+2}[9:0]</td>
</tr>
</tbody>
</table>

### 12-bit data

When EDM[1:0] = 10 in DCMI_CR, the camera interface captures the 14-bit data at its input DCMI_D[13:0] and stores them as the 14 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

### Table 251. Positioning of captured data bytes in 32-bit words (12-bit width)

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:28</th>
<th>27:16</th>
<th>15:12</th>
<th>11:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>D_{n+1}[11:0]</td>
<td>0</td>
<td>D_{n}[11:0]</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>D_{n+3}[11:0]</td>
<td>0</td>
<td>D_{n+2}[11:0]</td>
</tr>
</tbody>
</table>

### 14-bit data

When EDM[1:0] = 11 in DCMI_CR, the camera interface captures the 16-bit data at its input DCMI_D[15:0] and stores them as the 16 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.
34.3.6 DCMI synchronization

The digital camera interface supports embedded or hardware (DCMI_HSYNC and DCMI_VSYNC) synchronization. When embedded synchronization is used, it is up to the digital camera module to make sure that the 0x00 and 0xFF values are used ONLY for synchronization (not in data). Embedded synchronization codes are supported only for the 8-bit parallel data interface width (that is, in the DCMI_CR register, the EDM[1:0] bits must be cleared).

For compressed data, the DCMI supports only the hardware synchronization mode. In this case, DCMI_VSYNC is used as a start/end of the image, and DCMI_HSYNC is used as a Data Valid signal. Figure 259 shows the corresponding timing diagram.

### Table 252. Positioning of captured data bytes in 32-bit words (14-bit width)

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:30</th>
<th>29:16</th>
<th>15:14</th>
<th>13:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>D_n [13:0]</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>D_n+3 [13:0]</td>
<td>0</td>
<td>D_n+2 [13:0]</td>
</tr>
</tbody>
</table>

### Hardware synchronization mode

In hardware synchronization mode, the two synchronization signals (DCMI_HSYNC/DCMI_VSYNC) are used. Depending on the camera module/mode, data may be transmitted during horizontal/vertical synchronization periods. The DCMI_HSYNC/DCMI_VSYNC signals act like blanking signals since all the data received during DCMI_HSYNC/DCMI_VSYNC active periods are ignored.

In order to correctly transfer images into the DMA/RAM buffer, data transfer is synchronized with the DCMI_VSYNC signal. When the hardware synchronization mode is selected, and...
capture is enabled (CAPTURE bit set in DCMI_CR), data transfer is synchronized with the
deactivation of the DCMI_VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by DMA to successive
buffers or the same/circular buffer. To allow the DMA management of successive frames, a
VSIF (Vertical synchronization interrupt flag) is activated at the end of each frame.

**Embedded data synchronization mode**

In this synchronization mode, the data flow is synchronized using 32-bit codes embedded in
the data flow. These codes use the 0x00/0xFF values that are not used in data anymore.
There are 4 types of codes, all with a 0xFF0000XY format. The embedded synchronization
codes are supported only in 8-bit parallel data width capture (in the DCMI_CR register, the
EDM[1:0] bits must be cleared). For other data widths, this mode generates unpredictable
results and must not be used.

Note: Camera modules can have 8 such codes (in interleaved mode). For this reason, the
interleaved mode is not supported by the camera interface (otherwise, every other
half-frame would be discarded).

- **Mode 2**
  - Four embedded codes signal the following events
    - Frame start (FS)
    - Frame end (FE)
    - Line start (LS)
    - Line end (LE)
  
  The XY values in the 0xFF0000XY format of the four codes are programmable (see
  Section 34.5.7: DCMI embedded synchronization code register (DCMI_ESCR)).

  A 0xFF value programmed as a “frame end” means that all the unused codes are
  interpreted as valid frame end codes.

  In this mode, once the camera interface has been enabled, the frame capture starts
  after the first occurrence of the frame end (FE) code followed by a frame start (FS)
  code.

- **Mode 1**
  - An alternative coding is the camera mode 1. This mode is ITU656 compatible.

  The codes signal another set of events:
  - SAV (active line) - line start
  - EAV (active line) - line end
  - SAV (blanking) - end of line during interframe blanking period
  - EAV (blanking) - end of line during interframe blanking period

  This mode can be supported by programming the following codes:
  - FS ≤ 0xFF
  - FE ≤ 0xFF
  - LS ≤ SAV (active)
  - LE ≤ EAV (active)

  An embedded unmask code is also implemented for frame/line start and frame/line end
codes. Using it, it is possible to compare only the selected unmasked bits with the
programmed code. A bit can therefore be selected to compare in the embedded code and
detect a frame/line start or frame/line end. This means that there can be different codes for
the frame/line start and frame/line end with the unmasked bit position remaining the same.

**Example**

FS = 0xA5  
Unmask code for FS = 0x10  
In this case the frame start code is embedded in the bit 4 of the frame start code.

### 34.3.7 DCMI capture modes

This interface supports two types of capture: snapshot (single frame) and continuous grab.

**Snapshot mode (single frame)**

In this mode, a single frame is captured (CM = 1 of the DCMI_CR register). After the
CAPTURE bit is set in DCMI_CR, the interface waits for the detection of a start of frame
before sampling the data. The camera interface is automatically disabled (CAPTURE bit
cleared in DCMI_CR) after receiving the first complete frame. An interrupt is generated
(IT_FRAME) if it is enabled.

In case of an overrun, the frame is lost and the CAPTURE bit is cleared.

**Continuous grab mode**

In this mode (CM bit = 0 in DCMI_CR), once the CAPTURE bit has been set in DCMI_CR,
the grabbing process starts on the next DCMI_VSYNC or embedded frame start depending
on the mode. The process continues until the CAPTURE bit is cleared in DCMI_CR. Once
the CAPTURE bit has been cleared, the grabbing process continues until the end of the
current frame.

![Frame capture waveforms in snapshot mode](ai15832b)
In continuous grab mode, the FCRC[1:0] bits in DCMI_CR can be configured to grab all pictures, every second picture or one out of four pictures to decrease the frame capture rate.

Note: In the hardware synchronization mode (ESS = 0 in DCMI_CR), the IT_VSYNC interrupt is generated (if enabled) even when CAPTURE = 0 in DCMI_CR so, to reduce the frame capture rate even further, the IT_VSYNC interrupt can be used to count the number of frames between 2 captures in conjunction with the Snapshot mode. This is not allowed by embedded data synchronization mode.

34.3.8 DCMI crop feature

With the crop feature, the camera interface can select a rectangular window from the received image. The start (upper left corner) coordinates and size (horizontal dimension in number of pixel clocks and vertical dimension in number of lines) are specified using two 32-bit registers (DCMI_CWSTRT and DCMI_CWSIZE). The size of the window is specified in number of pixel clocks (horizontal dimension) and in number of lines (vertical dimension).

These registers specify the coordinates of the starting point of the capture window as a line number (in the frame, starting from 0) and a number of pixel clocks (on the line, starting from 0), and the size of the window as a line number and a number of pixel clocks. The CAPCNT value can only be a multiple of 4 (two least significant bits are forced to 0) to allow the correct transfer of data through the DMA.
If the DCMI_VSYNC signal goes active before the number of lines is specified in the DCMI_CWSIZE register, then the capture stops and an IT_FRAME interrupt is generated when enabled.

**Figure 263. Data capture waveforms**

1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

### 34.3.9 DCMI JPEG format

To allow JPEG image reception, it is necessary to set the JPEG bit of the DCMI_CR register. JPEG images are not stored as lines and frames, so the DCMI_VSYNC signal is used to start the capture while DCMI_HSYNC serves as a data enable signal. The number of bytes in a line may not be a multiple of 4. This case must be carefully handled since a DMA request is generated each time a complete 32-bit word has been constructed from the captured data. When an end of frame is detected and the 32-bit word to be transferred has not been completely received, the remaining data are padded with zeros and a DMA request is generated.

The crop feature and embedded synchronization codes cannot be used in JPEG format.

### 34.3.10 DCMI FIFO

A 8-word FIFO is implemented to manage data rate transfers on the AHB. The DCMI features a simple FIFO controller with a read pointer incremented each time the camera interface reads from the AHB, and a write pointer incremented each time the camera interface writes to the FIFO. There is no overrun protection to prevent the data from being overwritten if the AHB interface does not sustain the data transfer rate.

In case of overrun or errors in the synchronization signals, the FIFO is reset and the DCMI interface waits for a new start of frame.
34.3.11 DCMI data format description

Data formats

Three types of data are supported:
- 8/10/12/14-bit progressive video: either monochrome or raw Bayer format
- YCbCr 4:2:2 progressive video
- RGB565 progressive video. A pixel coded in 16 bits (5 bits for blue, 5 bits for red, 6 bits for green) takes two clock cycles to be transferred.

Compressed data: JPEG

For B&W (black and white), YCbCr or RGB data, the maximum input size is 2048 × 2048 pixels. No limit in JPEG compressed mode.

For monochrome, RGB and YCbCr, the frame buffer is stored in raster mode. 32-bit words are used. Only the little-endian format is supported.

Figure 264. Pixel raster scan order

Monochrome format

Characteristics:
- Raster format
- 8 bits per pixel

The table below shows how the data are stored.

Table 253. Data storage in monochrome progressive video format

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:24</th>
<th>23:16</th>
<th>15:8</th>
<th>7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>n + 3</td>
<td>n + 2</td>
<td>n + 1</td>
<td>n</td>
</tr>
<tr>
<td>4</td>
<td>n + 7</td>
<td>n + 6</td>
<td>n + 5</td>
<td>n + 4</td>
</tr>
</tbody>
</table>

RGB format

Characteristics:
- Raster format
- RGB
- Interleaved: one buffer: R, G and B interleaved (such as BRGBRGRBGRG)
- Optimized for display output
The RGB planar format is compatible with standard OS frame buffer display formats. Only 16 BPP (bits per pixel): RGB565 (2 pixels per 32-bit word) is supported.

The 24 BPP (palletized format) and gray-scale formats are not supported. Pixels are stored in a raster scan order, that is from top to bottom for pixel rows, and from left to right within a pixel row. Pixel components are R (red), G (green) and B (blue). All components have the same spatial resolution (4:4:4 format). A frame is stored in a single part, with the components interleaved on a pixel basis.

The table below shows how the data are stored.

**Table 254. Data storage in RGB progressive video format**

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:27</th>
<th>26:21</th>
<th>20:16</th>
<th>15:11</th>
<th>10:5</th>
<th>4:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Red n + 1</td>
<td>Green n + 1</td>
<td>Blue n + 1</td>
<td>Red n</td>
<td>Green n</td>
<td>Blue n</td>
</tr>
<tr>
<td>4</td>
<td>Red n + 4</td>
<td>Green n + 3</td>
<td>Blue n + 3</td>
<td>Red n + 2</td>
<td>Green n + 2</td>
<td>Blue n + 2</td>
</tr>
</tbody>
</table>

**YCbCr format**

Characteristics:
- Raster format
- YCbCr 4:2:2
- Interleaved: one buffer: Y, Cb and Cr interleaved (such as CbYCrYCbYCr)

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). Each component is encoded in 8 bits. Luma and chroma are stored together (interleaved) as shown in the table below.

**Table 255. Data storage in YCbCr progressive video format**

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:24</th>
<th>23:16</th>
<th>15:8</th>
<th>7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Y n + 1</td>
<td>Cr n</td>
<td>Y n</td>
<td>Cb n</td>
</tr>
<tr>
<td>4</td>
<td>Y n + 3</td>
<td>Cr n + 2</td>
<td>Y n + 2</td>
<td>Cb n + 2</td>
</tr>
</tbody>
</table>

**YCbCr format - Y only**

Characteristics:
- Raster format
- YCbCr 4:2:2
- The buffer only contains Y information - monochrome image

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). In this mode, the chroma information is dropped. Only the luma component of each pixel, encoded in 8 bits, is stored as shown in Table 256.

The result is a monochrome image having the same resolution as the original YCbCr data.
Half resolution image extraction

This is a modification of the previous reception modes, being applicable to monochrome, RGB or Y extraction modes.

This mode is used to only store a half resolution image. It is selected through OELS and LSM control bits.

34.4 DCMI interrupts

Five interrupts are generated. All interrupts are maskable by software. The global interrupt (dcmi_it) is the OR of all the individual interrupts. The table below gives the list of all interrupts.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exits Sleep mode</th>
<th>Exists Stop and Standby modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcmi_it</td>
<td>End of line</td>
<td>LINE_RIS</td>
<td>LINE_IE</td>
<td>Set LINE_ISC</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>End of frame capture</td>
<td>FRAME_RIS</td>
<td>FRAME_IE</td>
<td>Set FRAME_ISC</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Overrun of data reception</td>
<td>OVR_RIS</td>
<td>OVR_IE</td>
<td>Set OVR_ISC</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Synchronization frame</td>
<td>VSYNC_RIS</td>
<td>VSYNC_IE</td>
<td>Set VSYNC_ISC</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Detection of an error in the embedded</td>
<td>ERR_RIS</td>
<td>ERR_IE</td>
<td>Set ERR_ISC</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>synchronization frame detection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
34.5 **DCMI registers**

Refer to *Section 1.2 on page 97* for list of abbreviations used in register descriptions. All DCMI registers must be accessed as 32-bit words, otherwise a bus error occurs.

### 34.5.1 DCMI control register (DCMI\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bits 31:21 Reserved, must be kept at reset value.

**Bit 20** **OELS**: Odd/Even Line Select (Line Select Start)

This bit works in conjunction with the LSM field (LSM = 1).

0: Interface captures first line after the frame start, second one being dropped.

1: Interface captures second line from the frame start, first one being dropped.

**Bit 19** **LSM**: Line Select mode

0: Interface captures all received lines.

1: Interface captures one line out of two.

**Bit 18** **OEBS**: Odd/Even Byte Select (Byte Select Start)

This bit works in conjunction with BSM field (BSM ≠ 00).

0: Interface captures first data (byte or double byte) from the frame/line start, second one being dropped.

1: Interface captures second data (byte or double byte) from the frame/line start, first one being dropped.

**Bits 17:16** **BSM\[1:0\]**: Byte Select mode

00: Interface captures all received data.

01: Interface captures every other byte from the received data.

10: Interface captures one byte out of four.

11: Interface captures two bytes out of four.

*Note: This mode only works for EDM\[1:0\] = 00. For all other EDM values, this field must be programmed to the reset value.*

**Bit 15** Reserved, must be kept at reset value.

**Bit 14** **ENABLE**: DCMI enable

0: DCMI disabled

1: DCMI enabled

*Note: The DCMI configuration registers must be programmed correctly before enabling this bit.*

**Bits 13:12** Reserved, must be kept at reset value.
Bits 11:10  **EDM[1:0]**: Extended data mode
- 00: Interface captures 8-bit data on every pixel clock.
- 01: Interface captures 10-bit data on every pixel clock.
- 10: Interface captures 12-bit data on every pixel clock.
- 11: Interface captures 14-bit data on every pixel clock.

Bits 9:8  **FCRC[1:0]**: Frame capture rate control
These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode. They are ignored in snapshot mode.
- 00: All frames are captured.
- 01: Every alternate frame captured (50% bandwidth reduction)
- 10: One frame out of four captured (75% bandwidth reduction)
- 11: reserved

Bit 7  **VSPOL**: Vertical synchronization polarity
This bit indicates the level on the DCMI_VSYNC pin when the data are not valid on the parallel interface.
- 0: DCMI_VSYNC active low
- 1: DCMI_VSYNC active high

Bit 6  **HSPOL**: Horizontal synchronization polarity
This bit indicates the level on the DCMI_HSYNC pin when the data are not valid on the parallel interface.
- 0: DCMI_HSYNC active low
- 1: DCMI_HSYNC active high

Bit 5  **PCKPOL**: Pixel clock polarity
This bit configures the capture edge of the pixel clock.
- 0: Falling edge active
- 1: Rising edge active

Bit 4  **ESS**: Embedded synchronization select
- 0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the DCMI_HSYNC/DCMI_VSYNC signals.
- 1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.
  *Note*: Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when the ESS bit is set. This bit is disabled in JPEG mode.

Bit 3  **JPEG**: JPEG format
- 0: Uncompressed video format
- 1: This bit is used for JPEG data transfers. The DCMI_HSYNC signal is used as data enable. The crop and embedded synchronization features (ESS bit) cannot be used in this mode.

Bit 2  **CROP**: Crop feature
- 0: The full image is captured. In this case the total number of bytes in an image frame must be a multiple of four.
- 1: Only the data inside the window specified by the crop register is captured. If the size of the crop window exceeds the picture size, then only the picture size is captured.

Bit 1  **CM**: Capture mode
- 0: Continuous grab mode - The received data are transferred into the destination memory through the DMA. The buffer location and mode (linear or circular buffer) is controlled through the system DMA.
- 1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame and then transfers a single frame through the DMA. At the end of the frame, the CAPTURE bit is automatically reset.
34.5.2 DCMI status register (DCMI_SR)

Address offset: 0x04
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-3</th>
<th>Bit 28-24</th>
<th>Bit 23-19</th>
<th>Bit 18-14</th>
<th>Bit 13-9</th>
<th>Bit 8-4</th>
<th>Bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>FNE</td>
<td>VSYNC</td>
<td>HSYNC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 2 **FNE**: FIFO not empty
This bit gives the status of the FIFO.
0: FIFO contains valid data.
1: FIFO empty

Bit 1 **VSYNC**: Vertical synchronization
This bit gives the state of the DCMI_VSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
0: active frame
1: synchronization between frames
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

Bit 0 **HSYNC**: Horizontal synchronization
This bit gives the state of the DCMI_HSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
0: active line
1: synchronization between lines
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

Note: The DMA controller and all DCMI configuration registers must be programmed correctly before enabling this bit.
34.5.3 DCMI raw interrupt status register (DCMI_RIS)

DCMI_RIS gives the raw interrupt status and is accessible in read only. When read, this register returns the status of the corresponding interrupt before masking with the DCMI_IER register value.

Address offset: 0x08
Reset value: 0x0000 0000

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

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

**Bits 31:5** Reserved, must be kept at reset value.

**Bit 4 LINE_RIS**: Line raw interrupt status
This bit gets set when the DCMI_HSYNC signal changes from the inactive state to the active state. It goes high even if the line is not valid.
In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMI_CR is set.
It is cleared by setting the LINE_ISC bit of the DCMI_ICR register.

**Bit 3 VSYNC_RIS**: DCMI_VSYNC raw interrupt status
This bit is set when the DCMI_VSYNC signal changes from the inactive state to the active state.
In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMI_CR.
It is cleared by setting the VSYNC_ISC bit of the DCMI_ICR register.

**Bit 2 ERR_RIS**: Synchronization error raw interrupt status
0: No synchronization error detected
1: Embedded synchronization characters are not received in the correct order.
This bit is valid only in the embedded synchronization mode. It is cleared by setting the ERR_ISC bit of the DCMI_ICR register.

*Note: This bit is available only in embedded synchronization mode.*

**Bit 1 OVR_RIS**: Overrun raw interrupt status
0: No data buffer overrun occurred
1: A data buffer overrun occurred and the data FIFO is corrupted.
The bit is cleared by setting the OVR_ISC bit of the DCMI_ICR register.

**Bit 0 FRAME_RIS**: Capture complete raw interrupt status
0: No new capture
1: A frame has been captured.
This bit is set when a frame or window has been captured.
In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (e.g. window cropped outside the frame).
The bit is cleared by setting the FRAME_ISC bit of the DCMI_ICR register.
34.5.4 DCMI interrupt enable register (DCMI_IER)

The DCMI_IER register is used to enable interrupts. When one of the DCMI_IER bits is set, the corresponding interrupt is enabled. This register is accessible in both read and write.

Address offset: 0x0C
Reset value: 0x0000 0000

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

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

Bits 31:5 Reserved, must be kept at reset value.

- **Bit 4 LINE_IE**: Line interrupt enable
  - 0: No interrupt generation when the line is received
  - 1: An Interrupt is generated when a line has been completely received.

- **Bit 3 VSYNC_IE**: DCMI_VSYNC interrupt enable
  - 0: No interrupt generation
  - 1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state.
  - The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

- **Bit 2 ERR_IE**: Synchronization error interrupt enable
  - 0: No interrupt generation
  - 1: An interrupt is generated if the embedded synchronization codes are not received in the correct order.
  - *Note*: This bit is available only in embedded synchronization mode.

- **Bit 1 OVR_IE**: Overrun interrupt enable
  - 0: No interrupt generation
  - 1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received.

- **Bit 0 FRAME_IE**: Capture complete interrupt enable
  - 0: No interrupt generation
  - 1: An interrupt is generated at the end of each received frame/crop window (in crop mode).
34.5.5 DCMI masked interrupt status register (DCMI_MIS)

This DCMI_MIS register is a read-only register. When read, it returns the current masked status value (depending on the value in DCMI_IER) of the corresponding interrupt. A bit in this register is set if the corresponding enable bit in DCMI_IER is set and the corresponding bit in DCMI_RIS is set.

Address offset: 0x10
Reset value: 0x0000 0000

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

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_MIS**: Line masked interrupt status
This bit gives the status of the masked line interrupt.
0: No interrupt generation when the line is received
1: An interrupt is generated when a line has been completely received and the LINE_IE bit is set in DCMI_IER.

Bit 3 **VSYNC_MIS**: VSYNC masked interrupt status
This bit gives the status of the masked VSYNC interrupt.
0: No interrupt is generated on DCMI_VSYNC transitions.
1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state and the VSYNC_IE bit is set in DCMI_IER.
The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_MIS**: Synchronization error masked interrupt status
This bit gives the status of the masked synchronization error interrupt.
0: No interrupt is generated on a synchronization error.
1: An interrupt is generated if the embedded synchronization codes are not received in the correct order and the ERR_IE bit in DCMI_IER is set.
*Note*: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_MIS**: Overrun masked interrupt status
This bit gives the status of the masked overflow interrupt.
0: No interrupt is generated on overrun.
1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received and the OVR_IE bit is set in DCMI_IER.

Bit 0 **FRAME_MIS**: Capture complete masked interrupt status
This bit gives the status of the masked capture complete interrupt
0: No interrupt is generated after a complete capture.
1: An interrupt is generated at the end of each received frame/crop window (in crop mode) and the FRAME_IE bit is set in DCMI_IER.
34.5.6  **DCMI interrupt clear register (DCMI_ICR)**

The DCMI_ICR register is write-only. Setting a bit of this register clears the corresponding flag in the DCMI_RIS and DCMI_MIS registers. Writing 0 has no effect.

Address offset: 0x14

Reset value: 0x0000 0000

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

Bits 31:5 Reserved, must be kept at reset value.

- **Bit 4** LINE_ISC: line interrupt status clear
  Setting this bit clears the LINE_RIS flag in the DCMI_RIS register.

- **Bit 3** VSYNC_ISC: Vertical Synchronization interrupt status clear
  Setting this bit clears the VSYNC_RIS flag in the DCMI_RIS register.

- **Bit 2** ERR_ISC: Synchronization error interrupt status clear
  Setting this bit clears the ERR_RIS flag in the DCMI_RIS register.
  
  *Note: This bit is available only in embedded synchronization mode.*

- **Bit 1** OVR_ISC: Overrun interrupt status clear
  Setting this bit clears the OVR_RIS flag in the DCMI_RIS register.

- **Bit 0** FRAME_ISC: Capture complete interrupt status clear
  Setting this bit clears the FRAME_RIS flag in the DCMI_RIS register.

34.5.7  **DCMI embedded synchronization code register (DCMI_ESCR)**

Address offset: 0x18

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEC[7:0]</td>
<td>LEC[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSC[7:0]</td>
<td>FSC[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:24 **FEC[7:0]**: Frame end delimiter code
This byte specifies the code of the frame end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FEC.
If FEC is programmed to 0xFF, all the unused codes (0xFF0000XY) are interpreted as frame end delimiters.

Bits 23:16 **LEC[7:0]**: Line end delimiter code
This byte specifies the code of the line end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bits 15:8 **LSC[7:0]**: Line start delimiter code
This byte specifies the code of the line start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bits 7:0 **FSC[7:0]**: Frame start delimiter code
This byte specifies the code of the frame start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FSC.
If FSC is programmed to 0xFF, no frame start delimiter is detected. But, the first occurrence of LSC after an FEC code is interpreted as a start of frame delimiter.
34.5.8  DCMI embedded synchronization unmask register (DCMI_ESUR)

Address offset: 0x1C
Reset value: 0x0000 0000

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

**FEU[7:0]**

Bits 31:24  **FEU[7:0]**: Frame end delimiter unmask
This byte specifies the mask to be applied to the code of the frame end delimiter.
0: The corresponding bit in the FEC byte in DCMI_ESCR is masked while comparing the frame end delimiter with the received data.
1: The corresponding bit in the FEC byte in DCMI_ESCR is compared while comparing the frame end delimiter with the received data.

**LEU[7:0]**

Bits 23:16  **LEU[7:0]**: Line end delimiter unmask
This byte specifies the mask to be applied to the code of the line end delimiter.
0: The corresponding bit in the LEC byte in DCMI_ESCR is masked while comparing the line end delimiter with the received data.
1: The corresponding bit in the LEC byte in DCMI_ESCR is compared while comparing the line end delimiter with the received data.

**LSU[7:0]**

Bits 15:8  **LSU[7:0]**: Line start delimiter unmask
This byte specifies the mask to be applied to the code of the line start delimiter.
0: The corresponding bit in the LSC byte in DCMI_ESCR is masked while comparing the line start delimiter with the received data.
1: The corresponding bit in the LSC byte in DCMI_ESCR is compared while comparing the line start delimiter with the received data.

**FSU[7:0]**

Bits 7:0  **FSU[7:0]**: Frame start delimiter unmask
This byte specifies the mask to be applied to the code of the frame start delimiter.
0: The corresponding bit in the FSC byte in DCMI_ESCR is masked while comparing the frame start delimiter with the received data.
1: The corresponding bit in the FSC byte in DCMI_ESCR is compared while comparing the frame start delimiter with the received data.

34.5.9  DCMI crop window start (DCMI_CWSTRT)

Address offset: 0x20
Reset value: 0x0000 0000

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

**VST[12:0]**

**HOFFCNT[13:0]**
34.5.10  DCMI crop window size (DCMI_CWSIZE)

Address offset: 0x24
Reset value: 0x0000 0000

| Bits 31:30 Reserved, must be kept at reset value. |
| Bits 29:16 VST[12:0]: Vertical start line count |
| The image capture starts with this line number. Previous line data are ignored. |
| 0x0000: line 1 |
| 0x0001: line 2 |
| 0x0002: line 3 |
| .... |

| Bits 15:14 Reserved, must be kept at reset value. |
| Bits 13:0 HOFFCNT[13:0]: Horizontal offset count |
| This value gives the number of pixel clocks to count before starting a capture. |

| Bits 31:30 Reserved, must be kept at reset value. |
| Bits 29:16 VLINE[13:0]: Vertical line count |
| This value gives the number of lines to be captured from the starting point. |
| 0x0000: 1 line |
| 0x0001: 2 lines |
| 0x0002: 3 lines |
| .... |

| Bits 15:14 Reserved, must be kept at reset value. |
| Bits 13:0 CAPCNT[13:0]: Capture count |
| This value gives the number of pixel clocks to be captured from the starting point on the same line. It value must corresponds to word-aligned data for different widths of parallel interfaces. |
| 0x0000 => 1 pixel |
| 0x0001 => 2 pixels |
| 0x0002 => 3 pixels |
| .... |

34.5.11  DCMI data register (DCMI_DR)

Address offset: 0x28
Reset value: 0x0000 0000

The digital camera Interface packages all the received data in 32-bit format before requesting a DMA transfer. A 8-word deep FIFO is available to leave enough time for DMA transfers and avoid DMA overrun conditions.
<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>BYTE3[7:0]: Data byte 3</td>
</tr>
<tr>
<td>23:16</td>
<td>BYTE2[7:0]: Data byte 2</td>
</tr>
<tr>
<td>15:8</td>
<td>BYTE1[7:0]: Data byte 1</td>
</tr>
<tr>
<td>7:0</td>
<td>BYTE0[7:0]: Data byte 0</td>
</tr>
</tbody>
</table>
### 34.5.12 DCMI register map

**Table 258. DCMI register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset name</th>
<th>Byte0</th>
<th>Byte1</th>
<th>Byte2</th>
<th>Byte3</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>DCMI.CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>0x04</td>
<td>DCMI.SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>0x08</td>
<td>DCMI.RIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>0x0C</td>
<td>DCMI.IER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>0x10</td>
<td>DCMI.MIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>0x14</td>
<td>DCMI.ICR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>0x18</td>
<td>DCMI.ESCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>0x1C</td>
<td>DCMI.ESUR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>0x20</td>
<td>DCMI.CWSTART</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>0x24</td>
<td>DCMI.CWSIZE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>0x28</td>
<td>DCMI.DR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0000000000000000</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
35  Parallel synchronous slave interface (PSSI)

The PSSI peripheral and the DCMI (digital camera interface) use the same circuitry. As a result, these two peripherals cannot be used at the same time: when using the PSSI, the DCMI registers cannot be accessed, and vice-versa.

In addition, the PSSI and the DCMI share the same alternate functions and interrupt vector (see Section 35.3.2: PSSI pins and internal signals).

35.1  Introduction

The PSSI is a generic synchronous 8/16-bit parallel data input/output slave interface. It enables the transmitter to send a data valid signal that indicates when the data is valid, and the receiver to output a flow control signal that indicates when it is ready to sample the data.

35.2  PSSI main features

The PSSI peripheral main features are the following:

- Slave mode operation
- 8-bit or 16-bit parallel data input or output
- 8-word (32-byte) FIFO
- Data enable (PSSI_DE) alternate function input and Ready (PSSI_RDY) alternate function output

When selected, these inputs can either enable the transmitter to indicate when the data is valid, allow the receiver to indicate when it is ready to sample the data, or both.

35.3  PSSI functional description

The PSSI is a synchronous parallel slave interface that can send or receive high-speed data flows. It consists of up to 16 data lines (PSSI_D[15:0]) plus a clock line (PSSI_PDCK). The clock polarity can be configured so that data can be captured or transmitted on either the clock rising or falling edge.

Usually, a general-purpose DMA channel is used to pass 32-bit packed data via the data register (PSSI_DR).

The data flow can either be continuous or synchronized by hardware using the optional PSSI_DE (Data enable), and PSSI_RDY (Ready) signals.

*Figure 265* shows the PSSI block diagram.
35.3.1 PSSI block diagram

Figure 265. PSSI block diagram

Figure 266. Top-level block diagram

35.3.2 PSSI pins and internal signals

The PSSI interface is composed of 19 pins, though nine signals are enough to transfer parallel data. Table 259 shows the PSSI pins.

When the PSSI ENABLE bit (bit 14 of PSSI CR) is set to 1, the alternate functions and the interrupt vector are associated with the PSSI. Otherwise, they are associated with the DCMI. The DCMI ENABLE bit (bit 15 of DCMI CR) and the PSSI ENABLE bit (bit 14 of PSSI CR) must not be set to 1 at the same time. As an example, if a GPIO is configured to use the alternate function PSSI_PDCK/DCMI_PIXCK, it is the PSSI_PDCK function which becomes active if PSSI_CR/ENABLE is set to 1.
Table 259. PSSI input/output pins

<table>
<thead>
<tr>
<th>PSSI signal name</th>
<th>DCMI signal it is shared with</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSSI_PDCK</td>
<td>DCMI_PIXCK</td>
<td>Input</td>
<td>Parallel data clock input</td>
</tr>
<tr>
<td>PSSI_D[15:0]</td>
<td>DCMI_D[13:0]</td>
<td>Input/output</td>
<td>Data output when transmitting, data input when receiving</td>
</tr>
<tr>
<td>PSSI_DE</td>
<td>DCMI_HSYNC</td>
<td>Input</td>
<td>Data enable signal: data valid signal when receiving or flow control signal when transmitting</td>
</tr>
<tr>
<td>PSSI_RDY</td>
<td>DCMI_VSYNC</td>
<td>Output</td>
<td>Ready signal: flow control signal when receiving or data valid signal when transmitting</td>
</tr>
</tbody>
</table>

Table 260 shows the PSSI internal input/output signals.

Table 260. PSSI internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pssi_it</td>
<td>Output</td>
<td>Interrupt</td>
</tr>
<tr>
<td>pssi_dma</td>
<td>Output</td>
<td>DMA request</td>
</tr>
<tr>
<td>pssi_hclk</td>
<td>Input</td>
<td>AHB clock</td>
</tr>
</tbody>
</table>

35.3.3 PSSI clock

The AHB clock frequency must be at least 2.5 times higher than the PSSI_PDCK frequency. At frequency ratios lower than 2.5, data might be corrupted or lost during transfers.

Data transfers are synchronous with PSSI_PDCK. The PSSI_PDCK polarity can be configured as follows, through CKPOL bit (bit 5 of PSSI_CR):

- When CKPOL = 0
  - Input pins are sampled on PSSI_PDCK falling edge
  - Output pins are driven on PSSI_PDCK rising edge
- When CKPOL = 1
  - Input pins are sampled on PSSI_PDCK rising edge
  - Output pins are driven on PSSI_PDCK falling edge

35.3.4 PSSI data management

Data direction

The direction of data transfers is configured through the OUTEN control bit (bit 31 of PSSI_CR):

- When OUTEN is cleared to 0 (default setting), the PSSI operates in receive mode and the data is input on the data pins.
- When OUTEN is set to 1, the peripheral operates in transmit mode and the data is output on the data pins.

OUTEN can be modified only when the ENABLE bit is cleared to 0.
Data register and DMA

Data are transferred from/to the FIFO using the PSSI_DR data register:

- In receive mode, data must be read from the FIFO by reading PSSI_DR.
- In transmit mode, data must be written to the FIFO by writing into PSSI_DR.

Word (32-bit) accesses to PSSI_DR and half-word (16-bit) accesses to PSSI_DR[15:0] are permitted in all modes. Byte (8-bit) accesses to PSSI_DR[7:0] are permitted only when the PSSI is configured to transfer 8 bits at a time (EDM=00 in the PSSI_CR register).

To reduce the load on the CPU, it is recommended to use the DMA to transfer data from/to the PSSI FIFO. When it is used, the DMA must be configured to transfer data via the PSSI_DR register. Using 32-bit transfers optimizes bandwidth and reduces the bus load. However, 8-bit and 16-bit transfers are also permitted.

To use the DMA, set the PSSI DMA enable bit (DMAEN in PSSI_CR) to 1 (default setting). When DMAEN is set to 1, a DMA transfer is initiated when the FIFO is ready for a 32-bit transfer (four valid bytes in receive mode or four empty bytes in transmit mode). As a result, in receive mode, no DMA transfers are initiated if there are three bytes or fewer in the FIFO, even if the DMA is configured to perform 8-bit transfers.

The RTT4B and RTT1B status bits (PSSI_SR) are useful when the CPU directly performs transfers to and from the FIFO. RTT4B set to 1 indicates that the FIFO is ready to transfer four bytes: at least four valid bytes in the FIFO in receive mode or at least four free bytes in transmit mode. RTT1B set to 1 indicates that the FIFO is ready to transfer one byte: at least one valid byte in the FIFO in receive mode or at least one free byte in transmit mode.

8-bit data

The PSSI parallel interface can transfer either 8-bit (using D[7:0]) or 16-bit data (using D[15:0]) depending on the EDM[1:0] control bits (bits 11:10 of PSSI_CR). If the 8-bit configuration is selected (EDM[1:0] set to 00), the unused D[15:0] pins can be used for GPIO or other functions.

When EDM[1:0] in PSSI_CR are programmed to 00, the interface transfers 8 bits using the D[7:0] pins. In this case, D[15:8] are not used and four PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant byte (bits 7:0) correspond to the first byte transferred, and the most-significant byte (bits 31:28) corresponds to the forth byte transferred. Table 261 illustrates the positioning of the data bytes in two 32-bit words.

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:24</th>
<th>23:16</th>
<th>15:8</th>
<th>7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D_{n+3}[7:0]</td>
<td>D_{n+2}[7:0]</td>
<td>D_{n+1}[7:0]</td>
<td>D_n[7:0]</td>
</tr>
<tr>
<td>4</td>
<td>D_{n+7}[7:0]</td>
<td>D_{n+6}[7:0]</td>
<td>D_{n+5}[7:0]</td>
<td>D_{n+4}[7:0]</td>
</tr>
</tbody>
</table>

Table 261. Positioning of captured data bytes in 32-bit words (8-bit width)
16-bit data

When EDM[1:0] in PSSI_CR are programmed to 11, the interface transfers 16 bits using the D[15:0] pins. In this case, two PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant half word (bits 15:0) correspond to the first half word transferred, and the most-significant half-word (bits 31:16) corresponds to the second half word transferred. Table 262 illustrates the positioning of the data in two 32-bit words.

Table 262. Positioning of captured data bytes in 32-bit words (16-bit width)

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:16</th>
<th>15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(D_{n+1}[15:0])</td>
<td>(D_n[15:0])</td>
</tr>
<tr>
<td>4</td>
<td>(D_{n+3}[15:0])</td>
<td>(D_{n+2}[15:0])</td>
</tr>
</tbody>
</table>

FIFO data buffer and error conditions

A eight-word FIFO helps improving performance and avoids overruns and underruns.

If the ready signal (PSSI_RDY) is disabled in receive mode, an overrun error is generated when a clock active edge occurs when the FIFO is full. In this case, the input data is lost.

If the data enable signal (PSSI_DE) is disabled in transmit mode, an underrun error is generated when a clock active edge occurs when the FIFO is empty. In this case, unpredictable data are output.

The OVR_RIS status bit indicates that either an overrun or an underrun occurred. An interrupt can be generated when these events occur.

35.3.5 PSSI optional control signals

Data Enable (PSSI_DE) alternate function input

The data enable signal, PSSI_DE, is an optional signal. It is driven by the data source/transmitter in order to indicate that the data is valid to be transferred during the current cycle. When PSSI_DE is inactive, it means that the data will not be or should not be sampled by the receiver at the next clock edge.

This alternate function signal can be enabled using the DERDYCFG (bits 20:18 of PSSI_CR) control bits. PSSI_DE polarity is configured through DEPOL control bit (bit 6 of PSSI_CR). PSSI_DE is active low when DEPOL is cleared to 0, and high when DEPOL is set to 1.

The direction of the PSSI_DE signal is defined by the OUTEN value. It is the same as the data direction.

If the PSSI_DE alternate function input is enabled (through DERDYCFG) in receive mode (OUTEN cleared to 0), the PSSI samples PSSI_DE on the same PSSI_PDCK edge as the one used for sampling the data (D[15:0]). If PSSI_DE is active, the sampled data is saved in the FIFO. Otherwise, the sampled data is considered invalid and discarded. The transmitting device can use PSSI_DE as a data valid signal, driving it inactive when the data in the current cycle is not valid. This flow control function allows avoiding underrun errors.
If the PSSI_DE alternate output function is enabled (through DERDYCFG) in transmit mode (OUTEN=1), the PSSI drives PSSI_DE on the same PSSI_PDCK edge that the one used to drive the data (D[15:0]). If a new 8 or 16-bit data (as programmed in the EDM[1:0] control bits in PSSI_CR) is available for transmission in the internal FIFO, this data is output on the data outputs (D[15:0]) and the PSSI_DE output becomes active on the current PSSI_PDCK edge. Otherwise (if the TX FIFO is empty), the D[15:0] outputs remains unchanged on the next clock edge and the PSSI_DE output becomes inactive.

Ready (PSSI_RDY) alternate function output

The ready signal, PSSI_RDY, is an optional signal. It is driven by the receiving device and indicates whether data is being accepted in the current cycle. When PSSI_RDY is inactive, it means that the data will not be or should not be sampled by the receiver at the next clock edge.

This alternate function signal can be enabled using the DERDYCFG control bits (bits 20:18 of PSSI_CR). PSSI_RDY polarity is configured through the RDYPOL control bit (bit 6 of PSSI_CR). PSSI_RDY is active low when RDYPOL is cleared to 0, and high when RDYPOL set to 1.

The direction of the PSSI_RDY signal is defined by the OUTEN (bit 31 of PSSI_CR). It is set in the opposite direction compared to the PSSI_DE and data signals.
If the PSSI_RDY alternate output function is enabled (through DERDYCFG) in receive mode (OUTEN=0), the PSSI drives PSSI_RDY one PSSI_PDCK half cycle after it samples the data (D[15:0]). If the FIFO has enough free space to receive more data, the PSSI drives the PSSI_RDY signal active. Otherwise, if the FIFO is full and cannot accept more data, the PSSI drives the PSSI_RDY signal inactive. The transmitting device must repeat the current data in the next cycle when it detects that PSSI_RDY is inactive. This flow control function allows the PSSI to avoid overrun errors when the system (via the DMA) is unable to keep up with the data flow.

Figure 269. Ready in receive mode waveform diagram (CKPOL=0)

If the PSSI_RDY alternate input function is enabled (through DERDYCFG) in transmit mode (OUTEN=1), the PSSI samples the PSSI_RDY signal on the opposite PSSI_PDCK edge to the one at which D[15:0] are driven. If the PSSI_RDY signal is inactive, the PSSI keeps the same data (D[15:0]) and PSSI_DE signals that valid data are available during the next PSSI_PDCK clock cycle. Otherwise, if PSSI_RDY signal is sampled as active, the next data from the TX FIFO (if available) is output on the data outputs (D[15:0]). If no new data are available in the TX FIFO, the PSSI keeps the data output values and outputs the PSSI_DE signal as inactive (if enabled).

The receiving device uses the PSSI_RDY to control the data flow and avoid overrun errors when the system (via the DMA) is unable to keep up with the data flow.

Bidirectional PSSI_DE/PSSI_RDY signal

A single pin can be used for both data enable (PSSI_DE) and ready (PSSI_RDY) functions if DEPOL and RDYPOL are both set to 1 and DERDYCFG is set to 111 or 100 in the PSSI_CR register. In this case, the GPIO corresponding to selected alternate function (PSSI_DE when DERDYCFG=111 or PSSI_RDY when DERDYCFG=100) must be configured as open-drain. The other device must also be configured to drive the line as open-drain, and a weak pull-up must be applied to the line.

The signal thus becomes bidirectional. If either the sender drives the line low (to indicate that the data is not valid) or the receiver drives the line low (to indicate that it is not sampling the current data), then both devices will know that the data is not being transferred in the current cycle.
35.4 PSSI interrupts

The PSSI generates only one interrupt (IT_OVR). It is consequently equivalent to the global interrupt (pssi_it). Refer to Table 263 for the list of interrupts.

The PSSI and the DCMI share the same interrupt vector. When the PSSI ENABLE bit (bit 14 of PSSI_CR) is set to 1, these interrupts are triggered by the PSSI. Otherwise, they are controlled by the DCMI.

The DCMI ENABLE bit (bit 14 of DCMI_CR) and PSSI ENABLE bit must not be set to 1 at the same time.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Shared with DCMI</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from low-power mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT_OVR</td>
<td>IT_OVR</td>
<td>indicates overrun in receive mode or underrun in transmit mode</td>
<td>OVR_RIS</td>
<td>OVR_IE</td>
<td>OVR_ISC</td>
<td>NA</td>
</tr>
</tbody>
</table>

Table 263. PSSI interrupt requests
### 35.5 PSSI registers

An 8-bit write or a 16-bit write operation to any PSSI register besides PSSI_DR will result in a bus error. 32-bit read and write operations are permitted.

#### 35.5.1 PSSI control register (PSSI_CR)

Address offset: 0x00

Reset value: 0x4000 0000

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

**Bit 31 OUTEN:** Data direction selection bit
- 0: Receive mode: data is input synchronously with PSSI_PDCK
- 1: Transmit mode: data is output synchronously with PSSI_PDCK

**Bit 30 DMAEN:** DMA enable bit
- 0: DMA transfers are disabled. The user application can directly access the PSSI_DR register when DMA transfers are disabled.
- 1: DMA transfers are enabled (default configuration). A DMA channel in the general-purpose DMA controller must be configured to perform transfers from/to PSSI_DR.

Bits 29:21 Reserved, must be kept at reset value.

**Bits 20:18 DERDYCFG:** Data enable and ready configuration
- 000: PSSI_DE and PSSI_RDY both disabled
- 001: Only PSSI_RDY enabled
- 010: Only PSSI_DE enabled
- 011: Both PSSI_RDY and PSSI_DE alternate functions enabled
- 100: Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_RDY pin (see [Bidirectional PSSI_DE/PSSI_RDY signal on page 1244](#))
- 101: Only PSSI_RDY function enabled, but mapped to PSSI_DE pin
- 110: Only PSSI_DE function enabled, but mapped to PSSI_RDY pin
- 111: Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_DE pin (see [Bidirectional PSSI_DE/PSSI_RDY signal on page 1244](#))

When the PSSI_RDY function is mapped to the PSSI_DE pin (settings 101 or 111), it is still the RDYPOL bit which determines its polarity. Similarly, when the PSSI_DE function is mapped to the PSSI_RDY pin (settings 110 or 111), it is still the DEPOL bit which determines its polarity.

Bits 17:15 Reserved, must be kept at reset value.
Bit 14 **ENABLE**: PSSI enable
0: PSSI disabled
1: PSSI enabled
The contents of the FIFO are flushed when ENABLE is cleared to 0.

*Note:* When ENABLE=1, the content of PSSI_CR must not be changed, except for the ENABLE bit itself. All configuration bits can change as soon as ENABLE changes from 0 to 1.
The DMA controller and all PSSI configuration registers must be programmed correctly before setting the ENABLE bit to 1.
The ENABLE bit and the DCMI ENABLE bit (bit 15 of DCMI_CR) must not be set to 1 at the same time.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:10 **EDM[1:0]**: Extended data mode
00: Interface captures 8-bit data on every parallel data clock
01: Reserved, must not be selected
10: Reserved, must not be selected
11: The interface captures 16-bit data on every parallel data clock

Bit 9 Reserved, must be kept at reset value.

Bit 8 **RDYPOL**: Ready (PSSI_RDY) polarity
This bit indicates the level on the PSSI_RDY pin when the data are not valid on the parallel interface.
0: PSSI_RDY active low (0 indicates that the receiver is ready to receive)
1: PSSI_RDY active high (1 indicates that the receiver is ready to receive)

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DEPOL**: Data enable (PSSI_DE) polarity
This bit indicates the level on the PSSI_DE pin when the data are not valid on the parallel interface.
0: PSSI_DE active low (0 indicates that data is valid)
1: PSSI_DE active high (1 indicates that data is valid)

Bit 5 **CKPOL**: Parallel data clock polarity
This bit configures the capture edge of the parallel clock or the edge used for driving outputs, depending on OUTEN.
0: Falling edge active for inputs or rising edge active for outputs
1: Rising edge active for inputs or falling edge active for outputs.

Bits 4:0 Reserved, must be kept at reset value.

### 35.5.2 PSSI status register (PSSI_SR)

Address offset: 0x04
Reset value: 0x0000 0000
### 35.5.3 PSSI raw interrupt status register (PSSI_RIS)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

- **Bit 3 RTT1B**: FIFO is ready to transfer one byte
  - 1: FIFO is ready for a one-byte (32-bit) transfer. In receive mode, this means that at least one valid data byte is in the FIFO. In transmit mode, this means that there is at least one byte free in the FIFO.
  - 0: FIFO is not ready for a 1-byte transfer

- **Bit 2 RTT4B**: FIFO is ready to transfer four bytes
  - 1: FIFO is ready for a four-byte (32-bit) transfer. In receive mode, this means that at least four valid data bytes are in the FIFO. In transmit mode, this means that there are at least four bytes free in the FIFO.
  - 0: FIFO is not ready for a four-byte transfer

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

**PSSI_RIS** gives the raw interrupt status. This register is read-only. When read, it returns the status of the corresponding interrupt before masking with the PSSI_IER register value.

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

- **Bit 1 OVR_RIS**: Data buffer overrun/underrun raw interrupt status
  - 0: No overrun/underrun occurred
  - 1: An overrun/underrun occurred: overrun in receive mode, underrun in transmit mode.
  - This bit is cleared by writing a 1 to the OVR_ISC bit in PSSI_ICR.

- **Bit 0**: Reserved, must be kept at reset value.
35.5.4 PSSI interrupt enable register (PSSI_IER)

Address offset: 0x0C
Reset value: 0x0000 0000

The PSSI_IER register is used to enable interrupts. When one of the PSSI_IER bits is set, the corresponding interrupt is enabled. This register is accessible both in read and write modes.

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

Bit 1 OVR_IE: Data buffer overrun/underrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if either an overrun or an underrun error occurred.

Bit 0 Reserved, must be kept at reset value.

35.5.5 PSSI masked interrupt status register (PSSI_MIS)

This PSSI_MIS register is read-only. When read, it returns the current masked status value of the corresponding interrupt (depending on the value in PSSI_IER). A bit in this register is set if the corresponding enable bit in PSSI_IER is set and the corresponding bit in PSSI_RIS is set.

Address offset: 0x10
Reset value: 0x0000 0000
35.5.6 PSSI interrupt clear register (PSSI_ICR)

Address offset: 0x14
Reset value: 0x0000 0000

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

The PSSI_ICR register is write-only. Writing a 1 into a bit of this register clears the corresponding bit in the PSSI_RIS and PSSI_MIS registers. Writing a 0 has no effect. Reading this register always gives zeros.

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

Bit 1 **OVR_ICC**: Data buffer overrun/underrun interrupt status clear
Writing this bit to 1 clears the OVR_RIS bit in PSSI_RIS.

Bit 0 Reserved, must be kept at reset value.
35.5.7 PSSI data register (PSSI_DR)

Address offset: 0x28
Reset value: 0x0000 0000

In receive mode (OUTEN=0), the DMA controller must read the received data from this register. Write operations to PSSI_DR result in an error response. When more bytes than the number of valid bytes are read in the FIFO, the invalid bytes return zeros.

In transmit mode (OUTEN=1), the DMA controller must write the data to be transmitted into this register. Read operations to PSSI_DR result in an error response.

32-bit, 16-bit, and 8-bit accesses are all supported for PSSI_DR. For instance, 16-bit read/write operations remove/add two bytes from/to the FIFO. However, 8-bit accesses are permitted only when the PSSI is configured to transfer 8 data bits at a time (EDM=00 in PSSI_CR). 8-bit accesses to PSSI_DR when EDM is not set to 0 result in an error response.

All accesses must include byte 0: 8-bit accesses must be performed to bits 7 to 0 and 16-bit accesses from bits 15 to 0. Accesses that do not include byte 0 result in an error response.

Accessing PSSI_DR when ENABLE bit in PSSI_CR is set to 0 results in an error response.

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

BYTE3

31:24 Bits 31:24 **BYTE3**: Data byte 3
23:16 Bits 23:16 **BYTE2**: Data byte 2
15:8 Bits 15:8 **BYTE1**: Data byte 1
7:0 Bits 7:0 **BYTE0**: Data byte 0
35.5.8 PSSI register map

*Table 264* summarizes the PSSI registers.

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | PSSI_CR  |    |    |    |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | PSSI_SR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | PSSI_RIS |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | PSSI_IER |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | PSSI_MIS |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x14   | PSSI_ICR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x18   | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x18   | to       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x24   |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x28   | PSSI_DR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |          |    |    |    |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x2C   | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2C   | to       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x3EC  |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to *Section 2.3 on page 128* for the register boundary addresses.
36  LCD-TFT display controller (LTDC)

36.1  Introduction

The LCD-TFT (liquid crystal display - thin film transistor) display controller provides a parallel digital RGB (red, green, blue) and signals for horizontal, vertical synchronization, pixel clock and data enable as output to interface directly to a variety of LCD and TFT panels.

36.2  LTDC main features

- 24-bit RGB parallel pixel output; 8 bits-per-pixel (RGB888)
- 2 display layers with dedicated FIFO (64x64-bit)
- Color look-up table (CLUT) up to 256 color (256x24-bit) per layer
- Programmable timings for different display panels
- Programmable background color
- Programmable polarity for HSYNC, VSYNC and data enable
- Up to 8 input color formats selectable per layer:
  - ARGB8888
  - RGB888
  - RGB565
  - ARGB1555
  - ARGB4444
  - L8 (8-bit luminance or CLUT)
  - AL44 (4-bit alpha + 4-bit luminance)
  - AL88 (8-bit alpha + 8-bit luminance)
- Pseudo-random dithering output for low bits per channel
  - Dither width 2 bits for red, green, blue
- Flexible blending between two layers using alpha value (per pixel or constant)
- Color keying (transparency color)
- Programmable window position and size
- Supports thin film transistor (TFT) color displays
- AXI master interface with burst of 16 double-words
- Up to 4 programmable interrupt events
36.3 LTDC functional description

36.3.1 LTDC block diagram

The block diagram of the LTDC is shown in the figure below.

**Figure 272. LTDC block diagram**

Layer FIFO: One FIFO 64x64-bit per layer.

PFC: pixel format converter, performing the pixel format conversion from the selected input pixel format of a layer to words.

AXI interface: for data transfer from memories to the FIFO.

Blending, dithering unit and timings generator: Refer to Section 36.4.1 and Section 36.4.2.

36.3.2 LCD-TFT internal signals

The table below gives the list of LCD-TFT internal signals.

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ltdc_aclk</td>
<td>Digital input</td>
<td>Clock for LCD-TFT registers in AXI clock domain</td>
</tr>
<tr>
<td>ltdc_pclk</td>
<td>Digital input</td>
<td>LCD-TFT register interface clock</td>
</tr>
<tr>
<td>ltdc_ker_ck</td>
<td>Digital input</td>
<td>LCD-TFT kernel clock used for LCD_CLK (pixel clock) generation</td>
</tr>
<tr>
<td>ltdc_li_it</td>
<td>Digital output</td>
<td>LCD-TFT line interrupt trigger for MDMA</td>
</tr>
</tbody>
</table>
### 36.3.3 LTDC pins and external signal interface

The table below summarizes the LTDC signal interface.

<table>
<thead>
<tr>
<th>LCD-TFT signals</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD_CLK</td>
<td>O</td>
<td>Clock output</td>
</tr>
<tr>
<td>LCD_HSYNC</td>
<td>O</td>
<td>Horizontal synchronization</td>
</tr>
<tr>
<td>LCD_VSYNC</td>
<td>O</td>
<td>Vertical synchronization</td>
</tr>
<tr>
<td>LCD_DE</td>
<td>O</td>
<td>Not data enable</td>
</tr>
<tr>
<td>LCD_R[7:0]</td>
<td>O</td>
<td>Data: 8-bit red data</td>
</tr>
<tr>
<td>LCD_G[7:0]</td>
<td>O</td>
<td>Data: 8-bit green data</td>
</tr>
<tr>
<td>LCD_B[7:0]</td>
<td>O</td>
<td>Data: 8-bit blue data</td>
</tr>
</tbody>
</table>

The LTDC-TFT controller pins must be configured by the user application. The unused pins can be used for other purposes.

For LTDC outputs up to 24 bits (RGB888), if less than 8 bpp are used to output for example RGB565 or RGB666 to interface on 16- or 18-bit displays, the RGB display data lines must be connected to the MSB of the LCD-TFT controller RGB data lines. As an example, in the case of an LCD-TFT controller interfacing with a RGB565 16-bit display, the LCD display R[4:0], G[5:0] and B[4:0] data lines pins must be connected to LCD-TFT controller LCD_R[7:3], LCD_G[7:2] and LCD_B[7:3].

### 36.3.4 LTDC reset and clocks

The LCD-TFT controller peripheral uses the following clock domains:

- AXI clock domain (ltdc_aclk)
  
  This domain contains the LCD-TFT AXI master interface for data transfer from the memories to the Layer FIFO and the frame buffer configuration register

- APB clock domain (ltdc_pclk):
  
  This domain contains the global configuration registers and the interrupt register.

- Pixel clock domain (LCD_CLK)
  
  This domain contains the pixel data generation, the layer configuration register as well as the LCD-TFT interface signal generator. The LCD_CLK output should be configured following the panel requirements. The LCD_CLK is generated from a specific PLL output (refer to the reset and clock control section).
The table below summarizes the clock domain for each register.

**Table 267. Clock domain for each register**

<table>
<thead>
<tr>
<th>LTDC register</th>
<th>Clock domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTDC_LxCR</td>
<td>ltdc_aclk</td>
</tr>
<tr>
<td>LTDC_LxCFBAR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxCFBLR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxCFBLNR</td>
<td></td>
</tr>
<tr>
<td>LTDC_SRCR</td>
<td>ltdc_pclk</td>
</tr>
<tr>
<td>LTDC_IER</td>
<td></td>
</tr>
<tr>
<td>LTDC_ISR</td>
<td></td>
</tr>
<tr>
<td>LTDC_ICR</td>
<td></td>
</tr>
<tr>
<td>LTDC_SSCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_BPCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_AWCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_TWCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_GCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_BCCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LipCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_CPSR</td>
<td></td>
</tr>
<tr>
<td>LTDC_CDSR</td>
<td>Pixel clock (LCD_CLK)</td>
</tr>
<tr>
<td>LTDC_LxWHPCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxWVPCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxCKCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxPFCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxCACR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxDCCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxBFCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxCLUTWR</td>
<td></td>
</tr>
</tbody>
</table>

Care must be taken while accessing the LTDC registers, the APB bus is stalled during the access for a given time period (refer to *Table 268*).
The LCD controller can be reset by setting the corresponding bit in the RCC_APBRSTR register. It resets the three clock domains.

36.4 LTDC programmable parameters

The LCD-TFT controller provides flexible configurable parameters. It can be enabled or disabled through the LTDC_GCR register.

36.4.1 LTDC global configuration parameters

Synchronous timings

*Figure 273* presents the configurable timing parameters generated by the synchronous timings generator block presented in the block diagram *Figure 272*. It generates the horizontal and vertical synchronization timings panel signals, the pixel clock and the data enable signals.
The HBP and HFP are respectively the horizontal back porch and front porch period.
The VBP and the VFP are respectively the vertical back porch and front porch period.

The LCD-TFT programmable synchronous timings are the following:

- **HSYNC and VSYNC width**: horizontal and vertical synchronization width, configured by programming a value of HSYNC width - 1 and VSYNC width - 1 in the LTDC_SSCR register.
- **HBP and VBP**: horizontal and vertical synchronization back porch width, configured by programming the accumulated value HSYNC width + HBP - 1 and the accumulated value VSYNC width + VBP - 1 in the LTDC_BPCR register.
- **Active width and active height**: the active width and active height are configured by programming the accumulated value HSYNC width + HBP + active width - 1 and the accumulated value VSYNC width + VBP + active height - 1 in the LTDC_AWCR register.
- **Total width**: the total width is configured by programming the accumulated value HSYNC width + HBP + active width + HFP - 1 in the LTDC_TWCR register. The HFP is the horizontal front porch period.
- **Total height**: the total height is configured by programming the accumulated value VSYNC height + VBP + active height + VFP - 1 in the LTDC_TWCR register. The VFP is the vertical front porch period.

**Note:** When the LTDC is enabled, the timings generated start with X/Y=0/0 position as the first horizontal synchronization pixel in the vertical synchronization area and following the back porch, active data display area and the front porch.
When the LTDC is disabled, the timing generator block is reset to \( X = \text{total width} - 1 \), \( Y = \text{total height} - 1 \) and held the last pixel before the vertical synchronization phase and the FIFO are flushed. Therefore only blanking data is output continuously.

**Example of synchronous timings configuration**

LCD-TFT timings (must be extracted from panel datasheet):
- horizontal and vertical synchronization width: 0xA pixels and 0x2 lines
- horizontal and vertical back porch: 0x14 pixels and 0x2 lines
- active width and active height: 0x140 pixels, 0xF0 lines (320x240)
- horizontal front porch: 0xA pixels
- vertical front porch: 0x4 lines

The programmed values in the LTDC timings registers are:
- LTDC_SSCR register: to be programmed to 0x00090001. (HSW[11:0] is 0x9 and VSH[10:0] is 0x1)
- LTDC_BPCR register: to be programmed to 0x001D0003 (AHBP[11:0] is 0x1D(0xA+0x13) and AVBP[10:0]A is 0x3(0x2 + 0x1)).
- LTDC_AWCR register: to be programmed to 0x015D00F3 (AAW[11:0] is 0x15D(0xA+0x14 +0x13F) and AAH[10:0] is 0xF3(0x2 +0x2 + 0xEF).
- LTDC_TWCR register: to be programmed to 0x00000167 (TOTALW[11:0] is 0x167(0xA+0x14 +0x140 + 0x9).
- LTDC_THCR register: to be programmed to 0x000000F7 (TOTALH[10:0]is 0xF7(0x2 +0x2 + 0xF0 + 3)

**Programmable polarity**

The horizontal and vertical synchronization, data enable and pixel clock output signals polarity can be programmed to active high or active low through the LTDC_GCR register.

**Background color**

A constant background color (RGB888) can programmed through the LTDC_BCCR register. It is used for blending with the bottom layer.

**Dithering**

The dithering pseudo-random technique using an LFSR is used to add a small random value (threshold) to each pixel color channel (R, G or B) value, thus rounding up the MSB in some cases when displaying a 24-bit data on 18-bit display. Thus the dithering technique is used to round data which is different from one frame to the other.

The dithering pseudo-random technique is the same as comparing LSBs against a threshold value and adding a 1 to the MSB part only, if the LSB part is ≥ the threshold. The LSBs are typically dropped once dithering was applied.

The width of the added pseudo-random value is two bits for each color channel: two bits for red, two bits for green and two bits for blue.

Once the LCD-TFT controller is enabled, the LFSR starts running with the first active pixel and it is kept running even during blanking periods and when dithering is switched off. If the LTDC is disabled, the LFSR is reset.

The dithering can be switched on and off on the fly through the LTDC_GCR register.
Reload shadow registers

Some configuration registers are shadowed. The shadow registers values can be reloaded immediately to the active registers when writing to these registers or at the beginning of the vertical blanking period following the configuration in the LTDC_SRCLR register. If the immediate reload configuration is selected, the reload must be activated only when all new registers have been written.

The shadow registers must not be modified again before the reload is done. Reading from the shadow registers returns the actual active value. The new written value can only be read after the reload has taken place.

A register reload interrupt can be generated if enabled in the LTDCIER register.

The shadowed registers are all layer1 and layer2 registers except LTDC_LxCLUTWR.

Interrupt generation event

Refer to Section 36.5: LTDC interrupts for the interrupt configuration.

36.4.2 Layer programmable parameters

Up to two layers can be enabled, disabled and configured separately. The layer display order is fixed and it is bottom up. If two layers are enabled, the layer2 is the top displayed window.

Windowing

Every layer can be positioned and resized and it must be inside the active display area.

The window position and size are configured through the top-left and bottom-right X/Y positions and the internal timing generator that includes the synchronous, back porch size and the active data area. Refer to LTDC_LxWHPCR and LTDC_WVPCR registers.
The programmable layer position and size defines the first/last visible pixel of a line and the first/last visible line in the window. It allows to display either the full image frame or only a part of the image frame (see the figure below):

- The first and the last visible pixel in the layer are set by configuring the WHSTPOS[11:0] and WHSPPOS[11:0] in the LTDC_LxWHPCR register.
- The first and the last visible lines in the layer are set by configuring the WVSTPOS[10:0] and WVSPPOS[10:0] in the LTDC_LxWVPCR register.

**Figure 274. Layer window programmable parameters**

**Pixel input format**

The programmable pixel format is used for the data stored in the frame buffer of a layer. Up to height input pixel formats can be configured for every layer through the LTDC_LxPFCR register.

The pixel data is read from the frame buffer and then transformed to the internal 8888 (ARGB) format as follows: components having a width of less than 8 bits get expanded to 8 bits by bit replication. The selected bit range is concatenated multiple times until it is longer than 8 bits. Of the resulting vector, the 8 MSB bits are chosen. Example: 5 bits of an RGB565 red channel become (bit positions) 43210432 (the three LSBs are filled with the three MSBs of the five bits).

The table below describes the pixel data mapping depending on the selected format.

**Table 269. Pixel data mapping versus color format**

<table>
<thead>
<tr>
<th></th>
<th>ARGB8888</th>
<th>RGB888</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>@+3 Ax[7:0]</td>
<td>@+3 Bx[7:0]</td>
</tr>
<tr>
<td></td>
<td>@+2 Rx[7:0]</td>
<td>@+2 Gx[7:0]</td>
</tr>
<tr>
<td></td>
<td>@+1 Gx[7:0]</td>
<td>@+1 Gx[7:0]</td>
</tr>
<tr>
<td></td>
<td>Bx[7:0]</td>
<td>Bx[7:0]</td>
</tr>
<tr>
<td></td>
<td>@+6 Rx[7:0]</td>
<td>@+6 Gx[7:0]</td>
</tr>
<tr>
<td></td>
<td>@+5 Gx[7:0]</td>
<td>@+5 Gx[7:0]</td>
</tr>
<tr>
<td></td>
<td>Bx[7:0]</td>
<td>Bx[7:0]</td>
</tr>
<tr>
<td></td>
<td>@+4 Bx[7:0]</td>
<td>@+4 Bx[7:0]</td>
</tr>
<tr>
<td></td>
<td>Bx[7:0]</td>
<td>Bx[7:0]</td>
</tr>
</tbody>
</table>

The programmable pixel format is used for the data stored in the frame buffer of a layer. Up to height input pixel formats can be configured for every layer through the LTDC_LxPFCR register.

The pixel data is read from the frame buffer and then transformed to the internal 8888 (ARGB) format as follows: components having a width of less than 8 bits get expanded to 8 bits by bit replication. The selected bit range is concatenated multiple times until it is longer than 8 bits. Of the resulting vector, the 8 MSB bits are chosen. Example: 5 bits of an RGB565 red channel become (bit positions) 43210432 (the three LSBs are filled with the three MSBs of the five bits).

The table below describes the pixel data mapping depending on the selected format.
### Table 269. Pixel data mapping versus color format (continued)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>@+7</td>
<td>@+6</td>
<td>@+5</td>
<td>@+4</td>
<td></td>
</tr>
<tr>
<td>(G_{x+2}[7:0])</td>
<td>(B_{x+2}[7:0])</td>
<td>(R_{x+1}[7:0])</td>
<td>(G_{x+1}[7:0])</td>
<td></td>
</tr>
<tr>
<td><strong>RGB565</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@+3</td>
<td>@+2</td>
<td>@+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(R_{x+1}[4:0] G_{x+1}[5:3])</td>
<td>(G_{x+1}[2:0] B_{x+1}[4:0])</td>
<td>(R_{x}[4:0] G_{x}[5:3])</td>
<td>(G_{x}[2:0] B_{x}[4:0])</td>
<td></td>
</tr>
<tr>
<td>@+7</td>
<td></td>
<td>@+5</td>
<td>@+4</td>
<td></td>
</tr>
<tr>
<td>(R_{x+1}[4:0] G_{x+1}[5:3])</td>
<td>(G_{x+1}[2:0] B_{x+1}[4:0])</td>
<td>(R_{x+2}[4:0] G_{x+2}[5:3])</td>
<td>(G_{x+2}[2:0] B_{x+2}[4:0])</td>
<td></td>
</tr>
<tr>
<td><strong>ARGB1555</strong></td>
<td>@+3</td>
<td>@+2</td>
<td>@+1</td>
<td>@+4</td>
</tr>
<tr>
<td>(A_{x+1}[0] R_{x+1}[4:0] G_{x+1}[4:3])</td>
<td>(G_{x+1}[2:0] B_{x+1}[4:0])</td>
<td>(A_{x}[0] R_{x}[4:0] G_{x}[4:3])</td>
<td>(G_{x}[2:0] B_{x}[4:0])</td>
<td></td>
</tr>
<tr>
<td>@+7</td>
<td>@+6</td>
<td>@+5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(A_{x+3}[0] R_{x+3}[4:0] G_{x+3}[4:3])</td>
<td>(G_{x+3}[2:0] B_{x+3}[4:0])</td>
<td>(A_{x+2}[0] R_{x+2}[4:0] G_{x+2}[4:0])</td>
<td>(G_{x+2}[2:0] B_{x+2}[4:0])</td>
<td></td>
</tr>
<tr>
<td><strong>ARGB4444</strong></td>
<td>@+3</td>
<td>@+2</td>
<td>@+1</td>
<td>@</td>
</tr>
<tr>
<td>(A_{x+1}[3:0] R_{x+1}[3:0] G_{x+1}[3:0])</td>
<td>(G_{x+1}[3:0] B_{x+1}[3:0])</td>
<td>(A_{x}[3:0] R_{x}[3:0] G_{x}[3:0])</td>
<td>(G_{x}[3:0] B_{x}[3:0])</td>
<td></td>
</tr>
<tr>
<td>@+7</td>
<td>@+6</td>
<td>@+5</td>
<td>@+4</td>
<td></td>
</tr>
<tr>
<td>(A_{x+3}[3:0] R_{x+3}[3:0] G_{x+3}[3:0])</td>
<td>(G_{x+3}[3:0] B_{x+3}[3:0])</td>
<td>(A_{x+2}[3:0] R_{x+2}[3:0] G_{x+2}[3:0])</td>
<td>(G_{x+2}[3:0] B_{x+2}[3:0])</td>
<td></td>
</tr>
<tr>
<td><strong>L8</strong></td>
<td>@+3</td>
<td>@+2</td>
<td>@+1</td>
<td>@</td>
</tr>
<tr>
<td>(L_{x+3}[7:0])</td>
<td>(L_{x+2}[7:0])</td>
<td>(L_{x+1}[7:0])</td>
<td>(L_{x}[7:0])</td>
<td></td>
</tr>
<tr>
<td>@+7</td>
<td>@+6</td>
<td>@+5</td>
<td>@+4</td>
<td></td>
</tr>
<tr>
<td>(L_{x+7}[7:0])</td>
<td>(L_{x+6}[7:0])</td>
<td>(L_{x+5}[7:0])</td>
<td>(L_{x+4}[7:0])</td>
<td></td>
</tr>
<tr>
<td><strong>AL44</strong></td>
<td>@+3</td>
<td>@+2</td>
<td>@+1</td>
<td>@</td>
</tr>
<tr>
<td>(A_{x+3}[3:0] L_{x+3}[3:0])</td>
<td>(A_{x+2}[3:0] L_{x+2}[3:0])</td>
<td>(A_{x+1}[3:0] L_{x+1}[3:0])</td>
<td>(A_{x}[3:0] L_{x}[3:0])</td>
<td></td>
</tr>
<tr>
<td>@+7</td>
<td>@+6</td>
<td>@+5</td>
<td>@+4</td>
<td></td>
</tr>
<tr>
<td>(A_{x+7}[3:0] L_{x+7}[3:0])</td>
<td>(A_{x+6}[3:0] L_{x+6}[3:0])</td>
<td>(A_{x+5}[3:0] L_{x+5}[3:0])</td>
<td>(A_{x+4}[3:0] L_{x+4}[3:0])</td>
<td></td>
</tr>
<tr>
<td><strong>AL88</strong></td>
<td>@+3</td>
<td>@+2</td>
<td>@+1</td>
<td>@</td>
</tr>
<tr>
<td>(A_{x+1}[7:0])</td>
<td>(A_{x}[7:0])</td>
<td>(A_{x}[7:0])</td>
<td>(L_{x}[7:0])</td>
<td></td>
</tr>
<tr>
<td>@+7</td>
<td>@+6</td>
<td>@+5</td>
<td>@+4</td>
<td></td>
</tr>
<tr>
<td>(A_{x+3}[7:0])</td>
<td>(A_{x+2}[7:0])</td>
<td>(A_{x+2}[7:0])</td>
<td>(L_{x+2}[7:0])</td>
<td></td>
</tr>
</tbody>
</table>
**Color look-up table (CLUT)**

The CLUT can be enabled at run-time for every layer through the LTDC_LxCR register and it is only useful in case of indexed color when using the L8, AL44 and AL88 input pixel format.

First, the CLUT must be loaded with the R, G and B values that replace the original R, G, B values of that pixel (indexed color). Each color (RGB value) has its own address that is the position within the CLUT.

The R, G and B values and their own respective address are programmed through the LTDC_LxCLUTWR register:

- In case of L8 and AL88 input pixel format, the CLUT must be loaded by 256 colors. The address of each color is configured in the CLUTADD bits in the LTDC_LxCLUTWR register.
- In case of AL44 input pixel format, the CLUT must be loaded by only 16 colors. The address of each color must be filled by replicating the 4-bit L channel to 8-bit as follows:
  - L0 (indexed color 0), at address 0x00
  - L1, at address 0x11
  - L2, at address 0x22
  - .....  
  - L15, at address 0xFF

**Color frame buffer address**

Every layer has a start address for the color frame buffer configured through the LTDC_LxCFBAR register.

When a layer is enabled, the data is fetched from the color frame buffer.

**Color frame buffer length**

Every layer has a total line length setting for the color frame buffer in bytes and a number of lines in the frame buffer configurable in the LTDC_LxCFBLR and LTDC_LxCFBLNR register respectively.

The line length and the number of lines settings are used to stop the prefetching of data to the layer FIFO at the end of the frame buffer:

- If it is set to less bytes than required, a FIFO underrun interrupt is generated if it has been previously enabled.
- If it is set to more bytes than actually required, the useless data read from the FIFO is discarded. The useless data is not displayed.

**Color frame buffer pitch**

Every layer has a configurable pitch for the color frame buffer, that is the distance between the start of one line and the beginning of the next line in bytes. It is configured through the LTDC_LxCFBLR register.
Layer blending

The blending is always active and the two layers can be blended following the blending factors configured through the LTDC_LxBFCR register.

The blending order is fixed and it is bottom up. If two layers are enabled, first the Layer1 is blended with the Background color, then the layer2 is blended with the result of blended color of layer1 and the background. Refer to the figure below.

**Figure 275. Blending two layers with background**

![Figure 275. Blending two layers with background](msv48123v1)

Default color

Every layer can have a default color in the format ARGB which is used outside the defined layer window or when a layer is disabled.

The default color is configured through the LTDC_LxDCCR register.

The blending is always performed between the two layers even when a layer is disabled. To avoid displaying the default color when a layer is disabled, keep the blending factors of this layer in the LTDC_LxBFCR register to their reset value.

Color keying

A color key (RGB) can be configured to be representative for a transparent pixel.

If the color keying is enabled, the current pixels (after format conversion and before CLUT respectively blending) are compared to the color key. If they match for the programmed RGB value, all channels (ARGB) of that pixel are set to 0.

The color key value can be configured and used at run-time to replace the pixel RGB value.

The color keying is enabled through the LTDC_LxCKCR register.

The color keying is configured through the LTDC_LxCKCR register. The programmed value depends on the pixel format as it is compared to current pixel after pixel format conversion to ARGB888.

**Example:** if the a mid-yellow color (50 % red + 50 % green) is used as the transparent color key:

- In RGB565, the mid-yellow color is 0x8400. Set the LTDC_LxCKCR to 0x848200.
- In ARGB8888, the mid-yellow color is 0x808000. Set LTDC_LxCKCR to 0x808000.
- In all CLUT-based color modes (L8, AL88, AL44), set one of the palette entry to the mid-yellow color 0x808000 and set the LTDC_LxCKCR to 0x808000.
36.5 LTDC interrupts

The LTDC provides four maskable interrupts logically ORed to two interrupt vectors.

The interrupt sources can be enabled or disabled separately through the LTDC_IER register. Setting the appropriate mask bit to 1 enables the corresponding interrupt.

The two interrupts are generated on the following events:
- Line interrupt: generated when a programmed line is reached. The line interrupt position is programmed in the LTDC_LIPCR register
- Register reload interrupt: generated when the shadow registers reload is performed during the vertical blanking period
- FIFO underrun interrupt: generated when a pixel is requested from an empty layer FIFO
- Transfer error interrupt: generated when an AXI bus error occurs during data transfer

Those interrupts events are connected to the NVIC controller as described in the figure below.

![Figure 276. Interrupt events](image)

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>LIF</td>
<td>LIE</td>
</tr>
<tr>
<td>Register reload</td>
<td>RRIF</td>
<td>RRIEN</td>
</tr>
<tr>
<td>FIFO underrun</td>
<td>FUDERRIF</td>
<td>FUDERRIE</td>
</tr>
<tr>
<td>Transfer error</td>
<td>TERRIF</td>
<td>TERRIE</td>
</tr>
</tbody>
</table>
36.6 LTDC programming procedure

The steps listed below are needed to program the LTDC:

1. Enable the LTDC clock in the RCC register.
2. Configure the required pixel clock following the panel datasheet.
3. Configure the synchronous timings: VSYNC, HSYNC, vertical and horizontal back porch, active data area and the front porch timings following the panel datasheet as described in the Section 36.4.1: LTDC global configuration parameters.
4. Configure the synchronous signals and clock polarity in the LTDC_GCR register.
5. If needed, configure the background color in the LTDC_BCCR register.
6. Configure the needed interrupts in the LTDC_IER and LTDC_LIPCR register.
7. Configure the layer1/2 parameters by:
   - programming the layer window horizontal and vertical position in the LTDC_LxWHPCR and LTDC_WVPCR registers. The layer window must be in the active data area.
   - programming the pixel input format in the LTDC_LxPFCR register
   - programming the color frame buffer start address in the LTDC_LxCFBAR register
   - programming the line length and pitch of the color frame buffer in the LTDC_LxCFBLR register
   - programming the number of lines of the color frame buffer in the LTDC_LxCFBLNR register
   - if needed, loading the CLUT with the RGB values and its address in the LTDC_LxCFCLUTWR register
   - If needed, configuring the default color and the blending factors respectively in the LTDC_LxDCCR and LTDC_LxBFCR registers
8. Enable layer1/2 and if needed the CLUT in the LTDC_LxCR register.
9. If needed, enable dithering and color keying respectively in the LTDC_GCR and LTDC_LxCKCR registers. They can be also enabled on the fly.
10. Reload the shadow registers to active register through the LTDC_SRCR register.
11. Enable the LCD-TFT controller in the LTDC_GCR register.
12. All layer parameters can be modified on the fly except the CLUT. The new configuration must be either reloaded immediately or during vertical blanking period by configuring the LTDC_SRCR register.

Note: All layer’s registers are shadowed. Once a register is written, it must not be modified again before the reload has been done. Thus, a new write to the same register overrides the previous configuration if not yet reloaded.
36.7 LTDC registers

36.7.1 LTDC synchronization size configuration register (LTDC_SSCR)

This register defines the number of horizontal synchronization pixels minus 1 and the number of vertical synchronization lines minus 1. Refer to Figure 273 and Section 36.4: LTDC programmable parameters for an example of configuration.

Address offset: 0x08

Reset value: 0x0000 0000

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

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

Bits 27:16 **HSW[11:0]**: horizontal synchronization width (in units of pixel clock period)

These bits define the number of Horizontal Synchronization pixel minus 1.

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

Bits 10:0 **VSH[10:0]**: vertical synchronization height (in units of horizontal scan line)

These bits define the vertical Synchronization height minus 1. It represents the number of horizontal synchronization lines.

36.7.2 LTDC back porch configuration register (LTDC_BPCR)

This register defines the accumulated number of horizontal synchronization and back porch pixels minus 1 (HSYNC width + HBP - 1) and the accumulated number of vertical synchronization and back porch lines minus 1 (VSYNC height + VBP - 1). Refer to Figure 273 and Section 36.4: LTDC programmable parameters for an example of configuration.

Address offset: 0x0C

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>
36.7.3 LTDC active width configuration register (LTDC_AWCR)

This register defines the accumulated number of horizontal synchronization, back porch and active pixels minus 1 (HSYNC width + HBP + active width - 1) and the accumulated number of vertical synchronization, back porch lines and active lines minus 1 (VSYNC height + BVBP + active height - 1). Refer to Figure 273 and Section 36.4: LTDC programmable parameters for an example of configuration.

Address offset: 0x10
Reset value: 0x0000 0000

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

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

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

Bits 27:16 **AAW[11:0]:** accumulated active width (in units of pixel clock period)
These bits define the accumulated active width which includes the horizontal synchronization, horizontal back porch and active pixels minus 1. The active width is the number of pixels in active display area of the panel scan line. Refer to device datasheet for maximum active width supported following maximum pixel clock.

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

Bits 10:0 **AAH[10:0]:** accumulated active height (in units of horizontal scan line)
These bits define the accumulated height which includes the vertical synchronization, vertical back porch and the active height lines minus 1. The active height is the number of active lines in the panel. Refer to device datasheet for maximum active height supported following maximum pixel clock.
36.7.4 **LTDC total width configuration register (LTDC_TWCR)**

This register defines the accumulated number of horizontal synchronization, back porch, active and front porch pixels minus 1 (HSYNC width + HBP + active width + HFP - 1) and the accumulated number of vertical synchronization, back porch lines, active and front lines minus 1 (VSYNC height + BVBP + active height + VFP - 1). Refer to Figure 273 and Section 36.4: LTDC programmable parameters for an example of configuration.

Address offset: 0x14

Reset value: 0x0000 0000

| Bits 31:28 | Reserved, must be kept at reset value. |
| Bits 27:16 | **TOTALW[11:0]**: total width (in units of pixel clock period) |
| These bits defines the accumulated total width which includes the horizontal synchronization, horizontal back porch, active width and horizontal front porch pixels minus 1. |
| Bits 15:11 | Reserved, must be kept at reset value. |
| Bits 10:0 | **TOTALH[10:0]**: total height (in units of horizontal scan line) |
| These bits defines the accumulated height which includes the vertical synchronization, vertical back porch, the active height and vertical front porch height lines minus 1. |

36.7.5 **LTDC global control register (LTDC_GCR)**

This register defines the global configuration of the LCD-TFT controller.

Address offset: 0x18

Reset value: 0x0000 2220
Bit 31  **HSPOL**: horizontal synchronization polarity  
This bit is set and cleared by software.  
0: horizontal synchronization polarity is active low.  
1: horizontal synchronization polarity is active high.

Bit 30  **VSPOL**: vertical synchronization polarity  
This bit is set and cleared by software.  
0: vertical synchronization is active low.  
1: vertical synchronization is active high.

Bit 29  **DEPOL**: not data enable polarity  
This bit is set and cleared by software.  
0: not data enable polarity is active low.  
1: not data enable polarity is active high.

Bit 28  **PCPOL**: pixel clock polarity  
This bit is set and cleared by software.  
0: pixel clock polarity is active low.  
1: pixel clock is active high.

Bits 27:17  Reserved, must be kept at reset value.

Bit 16  **DEN**: dither enable  
This bit is set and cleared by software.  
0: dither disable  
1: dither enable

Bit 15  Reserved, must be kept at reset value.

Bits 14:12  **DRW[2:0]**: dither red width  
These bits return the Dither Red Bits.

Bit 11  Reserved, must be kept at reset value.

Bits 10:8  **DGW[2:0]**: dither green width  
These bits return the dither green bits.

Bit 7  Reserved, must be kept at reset value.

Bits 6:4  **DBW[2:0]**: dither blue width  
These bits return the dither blue bits.

Bits 3:1  Reserved, must be kept at reset value.

Bit 0  **LTDCEN**: LCD-TFT controller enable  
This bit is set and cleared by software.  
0: LTDC disable  
1: LTDC enable
36.7.6 LTDC shadow reload configuration register (LTDC_SRCR)

This register allows to reload either immediately or during the vertical blanking period, the shadow registers values to the active registers. The shadow registers are all Layer1 and Layer2 registers except the LTDC_L1CLUTWR and the LTDC_L2CLUTWR.

Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:2 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1 VBR: vertical blanking reload</td>
</tr>
<tr>
<td>This bit is set by software and cleared only by hardware after reload (it cannot be cleared through register write once it is set).</td>
</tr>
<tr>
<td>0: no effect</td>
</tr>
<tr>
<td>1: The shadow registers are reloaded during the vertical blanking period (at the beginning of the first line after the active display area).</td>
</tr>
<tr>
<td>Bit 0 IMR: immediate reload</td>
</tr>
<tr>
<td>This bit is set by software and cleared only by hardware after reload.</td>
</tr>
<tr>
<td>0: no effect</td>
</tr>
<tr>
<td>1: The shadow registers are reloaded immediately.</td>
</tr>
</tbody>
</table>

Note: The shadow registers read back the active values. Until the reload has been done, the 'old' value is read.

36.7.7 LTDC background color configuration register (LTDC_BCCR)

This register defines the background color (RGB888).
Address offset: 0x2C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:2 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1 VBR: vertical blanking reload</td>
</tr>
<tr>
<td>This bit is set by software and cleared only by hardware after reload.</td>
</tr>
<tr>
<td>0: no effect</td>
</tr>
<tr>
<td>1: The shadow registers are reloaded during the vertical blanking period (at the beginning of the first line after the active display area).</td>
</tr>
<tr>
<td>Bit 0 IMR: immediate reload</td>
</tr>
<tr>
<td>This bit is set by software and cleared only by hardware after reload.</td>
</tr>
<tr>
<td>0: no effect</td>
</tr>
<tr>
<td>1: The shadow registers are reloaded immediately.</td>
</tr>
</tbody>
</table>
36.7.8 LTDC interrupt enable register (LTDC_IER)

This register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

Address offset: 0x34
Reset value: 0x0000 0000

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

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

Bit 3 **RRIE**: register reload interrupt enable
This bit is set and cleared by software.
0: register reload interrupt disable
1: register reload interrupt enable

Bit 2 **TERRIE**: transfer error interrupt enable
This bit is set and cleared by software.
0: transfer error interrupt disable
1: transfer error interrupt enable

Bit 1 **FUIE**: FIFO underrun interrupt enable
This bit is set and cleared by software.
0: FIFO underrun interrupt disable
1: FIFO underrun interrupt enable

Bit 0 **LIE**: line interrupt enable
This bit is set and cleared by software.
0: line interrupt disable
1: line interrupt enable
### 36.7.9 LTDC interrupt status register (LTDC_ISR)

This register returns the interrupt status flag.

Address offset: 0x38  
Reset value: 0x0000 0000

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

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

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

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

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

- **Bit 3** **RRIF**: register reload interrupt flag  
  0: no register reload interrupt generated  
  1: register reload interrupt generated when a vertical blanking reload occurs (and the first line after the active area is reached)

- **Bit 2** **TERRIF**: transfer error interrupt flag  
  0: no transfer error interrupt generated  
  1: transfer error interrupt generated when a bus error occurs

- **Bit 1** **FUIF**: FIFO underrun interrupt flag  
  0: no FIFO underrun interrupt generated  
  1: FIFO underrun interrupt generated, if one of the layer FIFOs is empty and pixel data is read from the FIFO

- **Bit 0** **LIF**: line interrupt flag  
  0: no line interrupt generated  
  1: line interrupt generated when a programmed line is reached

### 36.7.10 LTDC Interrupt Clear Register (LTDC_ICR)

Address offset: 0x3C  
Reset value: 0x0000 0000

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

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

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

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

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

- **Bit 3** **CRRIF**: custom register reload interrupt flag  
  0: no custom register reload interrupt generated  
  1: custom register reload interrupt generated when a vertical blanking reload occurs (and the first line after the active area is reached)

- **Bit 2** **CTERRIF**: custom transfer error interrupt flag  
  0: no custom transfer error interrupt generated  
  1: custom transfer error interrupt generated when a bus error occurs

- **Bit 1** **CFUIF**: custom FIFO underrun interrupt flag  
  0: no custom FIFO underrun interrupt generated  
  1: custom FIFO underrun interrupt generated, if one of the layer FIFOs is empty and pixel data is read from the FIFO

- **Bit 0** **CLIF**: custom line interrupt flag  
  0: no custom line interrupt generated  
  1: custom line interrupt generated when a programmed line is reached
36.7.11 LTDC line interrupt position configuration register (LTDC_LIPCR)

This register defines the position of the line interrupt. The line value to be programmed depends on the timings parameters. Refer to Figure 273.

Address offset: 0x40
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>LIPOS[10:0]</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>Reserve, must be kept at reset value.</td>
</tr>
</tbody>
</table>

36.7.12 LTDC current position status register (LTDC_CPSR)

Address offset: 0x44
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>r r r r r r r r r r r r r r</td>
</tr>
</tbody>
</table>
36.7.13 **LTDC current display status register (LTDC_CDSR)**

This register returns the status of the current display phase which is controlled by the HSYNC, VSYNC, and horizontal/vertical DE signals.

Example: if the current display phase is the vertical synchronization, the VSYNCS bit is set (active high). If the current display phase is the horizontal synchronization, the HSYNCS bit is active high.

Address offset: 0x48

Reset value: 0x0000 000F

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

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

- **Bit 3** HSYNCS: horizontal synchronization display status
  - 0: active low
  - 1: active high

- **Bit 2** VSYNCS: vertical synchronization display status
  - 0: active low
  - 1: active high

- **Bit 1** HDES: horizontal data enable display status
  - 0: active low
  - 1: active high

- **Bit 0** VDES: vertical data enable display status
  - 0: active low
  - 1: active high

*Note:* The returned status does not depend on the configured polarity in the LTDC_GCR register, instead it returns the current active display phase.

36.7.14 **LTDC layer x control register (LTDC_LxCR)**

Address offset: 0x84 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000
36.7.15  **LTDC layer x window horizontal position configuration register (LTDC_LxWHPCR)**

This register defines the horizontal position (first and last pixel) of the layer 1 or 2 window.

The first visible pixel of a line is the programmed value of AHBP[11:0] bits + 1 in the LTDC_BPCR register.

The last visible pixel of a line is the programmed value of AAW[11:0] bits in the LTDC_AWCR register.

Address offset: 0x88 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000
Example:

The LTDC_BPCR register is configured to 0x000E0005 (AHBP[11:0] is 0xE) and the LTDC_AWCR register is configured to 0x028E01E5 (AAW[11:0] is 0x28E). To configure the horizontal position of a window size of 630x460, with horizontal start offset of 5 pixels in the active data area:

1. layer window first pixel, WHSTPOS[11:0], must be programmed to 0x14 (0xE+1+0x5)
2. layer window last pixel, WHSPPOS[11:0], must be programmed to 0x28A.

36.7.16 LTDC layer x window vertical position configuration register (LTDC_LxWVPCR)

This register defines the vertical position (first and last line) of the layer1 or 2 window.

The first visible line of a frame is the programmed value of AVBP[10:0] bits + 1 in the register LTDC_BPCR register.

The last visible line of a frame is the programmed value of AAH[10:0] bits in the LTDC_AWCR register.

Address offset: 0x8C + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
</tr>
</tbody>
</table>

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

Bits 26:16 WVSPPOS[10:0]: window vertical stop position
These bits configure the last visible line of the layer window.

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

Bits 10:0 WVSTPOS[10:0]: window vertical start position
These bits configure the first visible line of the layer window.
WVSTPOS[10:0] must be ≤ AAH[10:0] bits (programmed in LTDC_AWCR register).
Example:
The LTDC_BPCR register is configured to 0x000E0005 (AVBP[10:0] is 0x5) and the LTDC_AWCR register is configured to 0x028E01E5 (AAH[10:0] is 0x1E5).

To configure the vertical position of a window size of 630x460, with vertical start offset of 8 lines in the active data area:
1. layer window first line: WVSTPOS[10:0] must be programmed to 0xE (0x5 + 1 + 0x8).
2. layer window last line: WVSTPOS[10:0] must be programmed to 0x1DA.

36.7.17 LTDC layer x color keying configuration register (LTDC_LxCKCR)

This register defines the color key value (RGB), that is used by the color keying.
Address offset: 0x90 + 0x80 * (x - 1), (x = 1 to 2)
Reset value: 0x0000 0000

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

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

Bits 31:24  Reserved, must be kept at reset value.
Bits 23:16  CKRED[7:0]: color key red value
Bits 15:8   CKGREEN[7:0]: color key green value
Bits 7:0    CKBLUE[7:0]: color key blue value

36.7.18 LTDC layer x pixel format configuration register (LTDC_LxPFCR)

This register defines the pixel format that is used for the stored data in the frame buffer of a layer. The pixel data is read from the frame buffer and then transformed to the internal format 8888 (ARGB).
Address offset: 0x94 + 0x80 * (x - 1), (x = 1 to 2)
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF[2:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
36.7.19  **LTDC layer x constant alpha configuration register (LTDC_LxCACR)**

This register defines the constant alpha value (divided by 255 by hardware), that is used in the alpha blending. Refer to LTDC_LxBFCR register.

Address offset: 0x98 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 00FF

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Bit 23:16</th>
<th>Bit 15:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF[2:0]</td>
<td>CONSTA[7:0]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Bits 2:0**  
PF[2:0]: pixel format

These bits configure the pixel format

- 000: ARGB8888
- 001: RGB888
- 010: RGB565
- 011: ARGB1555
- 100: ARGB4444
- 101: L8 (8-bit luminance)
- 110: AL44 (4-bit alpha, 4-bit luminance)
- 111: AL88 (8-bit alpha, 8-bit luminance)

**Example:**

- If the programmed constant alpha is 0xFF, the constant alpha value is 255 / 255 = 1.
36.7.20 LTDC layer x default color configuration register (LTDC_LxDCCR)

This register defines the default color of a layer in the format ARGB. The default color is used outside the defined layer window or when a layer is disabled. The reset value of 0x00000000 defines a transparent black color.

Address offset: 0x9C + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24</th>
<th>23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCGREEN[7:0]</td>
<td>DCBLUE[7:0]</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw</td>
<td>rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:24 DCALPHA[7:0]: default color alpha
These bits configure the default alpha value.

Bits 23:16 DCREDS[7:0]: default color red
These bits configure the default red value.

Bits 15:8 DCGREEN[7:0]: default color green
These bits configure the default green value.

Bits 7:0 DCBLUE[7:0]: default color blue
These bits configure the default blue value.
36.7.21 LTDC layer x blending factors configuration register (LTDC_LxBFCR)

This register defines the blending factors F1 and F2.

The general blending formula is: \( BC = BF1 \times C + BF2 \times Cs \)

- \( BC \) = blended color
- \( BF1 \) = blend factor 1
- \( C \) = current layer color
- \( BF2 \) = blend factor 2
- \( Cs \) = subjacent layers blended color

Address offset: \( 0xA0 + 0x80 \times (x - 1) \), \( x = 1 \) to \( 2 \)

Reset value: 0x0000 0607

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

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **BF1[2:0]**: blending factor 1

These bits select the blending factor F1.

000: reserved
001: reserved
010: reserved
011: reserved
100: constant alpha
101: reserved
110: pixel alpha x constant alpha
111: reserved

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

Bits 2:0 **BF2[2:0]**: blending factor 2

These bits select the blending factor F2

000: reserved
001: reserved
010: reserved
011: reserved
100: reserved
101: 1 - constant alpha
110: reserved
111: 1 - (pixel alpha x constant alpha)
Note: The constant alpha value, is the programmed value in the LxCACR register divided by 255 by hardware.

Example: Only layer1 is enabled, BF1 configured to constant alpha. BF2 configured to 1 - constant alpha. The constant alpha programmed in the LxCACR register is 240 (0xF0). Thus, the constant alpha value is 240/255 = 0.94. C: current layer color is 128. Cs: background color is 48. Layer1 is blended with the background color.

\[ BC = \text{constant alpha} \times C + (1 - \text{Constant Alpha}) \times Cs = 0.94 \times 128 + (1-0.94) \times 48 = 123. \]

36.7.22 LTDC layer x color frame buffer address register (LTDC_LxCFBAR)

This register defines the color frame buffer start address which has to point to the address where the pixel data of the top left pixel of a layer is stored in the frame buffer.

Address offset: 0xAC + 0x80 \times (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

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

CFBADD[31:16]

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

CFBADD[15:0]

 Bits 31:0 CFBADD[31:0]: color frame buffer start address
These bits define the color frame buffer start address.

36.7.23 LTDC layer x color frame buffer length register (LTDC_LxCFBLR)

This register defines the color frame buffer line length and pitch.

Address offset: 0xB0 + 0x80 \times (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res. Res. Res. CFBL[12:0]</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

 Bits 31:29 Reserved, must be kept at reset value.

 Bits 28:16 CFBP[12:0]: color frame buffer pitch in bytes
These bits define the pitch that is the increment from the start of one line of pixels to the start of the next line in bytes.

 Bits 15:13 Reserved, must be kept at reset value.
Example:
- A frame buffer having the format RGB565 (2 bytes per pixel) and a width of 256 pixels (total number of bytes per line is 256 * 2 = 512), where pitch = line length requires a value of 0x02000207 to be written into this register.
- A frame buffer having the format RGB888 (3 bytes per pixel) and a width of 320 pixels (total number of bytes per line is 320 * 3 = 960), where pitch = line length requires a value of 0x03C003C7 to be written into this register.

36.7.24 LTDC layer x color frame buffer line number register (LTDC_LxCFBLNR)

This register defines the number of lines in the color frame buffer.

Address offset: 0xB4 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

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

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 CFBLNBR[10:0]: frame buffer line number

These bits define the number of lines in the frame buffer that corresponds to the active high width.

Note: The number of lines and line length settings define how much data is fetched per frame for every layer. If it is configured to less bytes than required, a FIFO underrun interrupt will be generated if enabled.

The start address and pitch settings on the other hand define the correct start of every line in memory.
36.7.25  LTDC layer x CLUT write register (LTDC_LxCLUTWR)

This register defines the CLUT address and the RGB value.

Address offset: 0xC4 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

Note: The CLUT write register must be configured only during blanking period or if the layer is disabled. The CLUT can be enabled or disabled in the LTDC_LxCR register.

The CLUT is only meaningful for L8, AL44 and AL88 pixel format.
## 36.7.26 LTDC register map

The following table summarizes the LTDC registers.

### Table 271. LTDC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name (LTDC)</th>
<th>Value (reset)</th>
<th>Value (in-use)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0008</td>
<td>LTDC_SSCR</td>
<td>HSW[11:0]</td>
<td>VSH[10:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x000C</td>
<td>LTDC_BPCR</td>
<td>AHBP[11:0]</td>
<td>AVBP[10:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0010</td>
<td>LTDC_AWCR</td>
<td>AAW[11:0]</td>
<td>AAH[10:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0014</td>
<td>LTDC_TWCR</td>
<td>TOTALW[11:0]</td>
<td>TOTALH[10:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0018</td>
<td>LTDC_GCR</td>
<td>HSPOL</td>
<td>VSPOL</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0024</td>
<td>LTDC_SRRCR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x002C</td>
<td>LTDC_BCCR</td>
<td>BCRED[7:0]</td>
<td>BCGREEN[7:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0034</td>
<td>LTDC_IER</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0038</td>
<td>LTDC_ISR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x003C</td>
<td>LTDC_ICR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0040</td>
<td>LTDC_LIPCR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0044</td>
<td>LTDC_CPSR</td>
<td>CXPOS[15:0]</td>
<td>CYPOS[15:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0048</td>
<td>LTDC_CDSR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

**Description:**
- **LTDC_SSCR (Offset 0x0008):** LTDC Synchronization Control Register. Contains HSYNC, VSYNC, and other control signals.
- **LTDC_BPCR (Offset 0x000C):** LTDC Bus Programming Control Register. Controls the timing and synchronization of the bus.
- **LTDC_AWCR (Offset 0x0010):** LTDC Address Write Control Register. Controls the address write function.
- **LTDC_TWCR (Offset 0x0014):** LTDC Timing Write Control Register. Controls the timing of the display.
- **LTDC_GCR (Offset 0x0018):** LTDC Global Control Register. Contains control bits for various operations.
- **LTDC_SRRCR (Offset 0x0024):** LTDC Source Register Control Register. Controls the source register.
- **LTDC_BCCR (Offset 0x002C):** LTDC Bus Control Register. Controls the bus activities.
- **LTDC_IER (Offset 0x0034):** LTDC Interrupt Enable Register. Enables or disables various interrupts.
- **LTDC_ISR (Offset 0x0038):** LTDC Interrupt Status Register. Indicates the status of various interrupts.
- **LTDC_ICR (Offset 0x003C):** LTDC Interrupt Clear Register. Clears the status of various interrupts.
- **LTDC_LIPCR (Offset 0x0040):** LTDC Line Interface Programming Control Register. Controls the line interface.
- **LTDC_CPSR (Offset 0x0044):** LTDC Clock Programming Control Register. Controls the clock programming.
- **LTDC_CDSR (Offset 0x0048):** LTDC Clock Driver Select Register. Selects the clock driver.
# Table 271. LTDC register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 0x0084 | LTDC_L1CR     | Reset value    |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
| 0x0088 | LTDC_L1WHPCR  | WHSPPOS[11:0]  | WHSTPOS[11:0]  |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
| 0x008C | LTDC_L1WVPCR  | WVSPPOS[10:0]  | WVSTPOS[10:0]  |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
| 0x0090 | LTDC_L1CKCR   | CKRED[7:0]     | CKGREEN[7:0]   | CKBLUE[7:0]    |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
| 0x0094 | LTDC_L1PFCR   | PF[2:0]        |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
| 0x0098 | LTDC_L1CACR   | CONSTA[7:0]    |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
| 0x009C | LTDC_L1DCCR   | DCALPHA[7:0]   | DCRED[7:0]     | DCGREEN[7:0]   | DCBLUE[7:0]    |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
| 0x00A0 | LTDC_L1BFCR   | BF1[2:0]       | BF2[2:0]       |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
| 0x00AC | LTDC_L1CFBAR  | CFBADD[31:0]   |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
| 0x00B0 | LTDC_L1CFBLR  | CFBP[12:0]     | CFBL[12:0]     |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
| 0x00B4 | LTDC_L1CFBLNR | CFBLNBR[10:0]  |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
| 0x00C4 | LTDC_L1CLUTWR | CLUTADD[7:0]   | RED[7:0]       | GREEN[7:0]     | BLUE[7:0]      |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
| 0x0104 | LTDC_L2CR     |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
| 0x0108 | LTDC_L2WHPCR  | WHSPPOS[11:0]  | WHSTPOS[11:0]  |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |                |
Refer to *Section 2.3 on page 128* for the register boundary addresses.
37 JPEG codec (JPEG)

37.1 Introduction
The hardware 8-bit JPEG codec encodes uncompressed image data stream or decodes JPEG-compressed image data stream. It also fully manages JPEG headers.

37.2 JPEG codec main features
- High-speed fully-synchronous operation
- Configurable as encoder or decoder
- Single-clock-per-pixel encode/decode
- RGB, YCbCr, YCMK and BW (grayscale) image color space support
- 8-bit depth per image component at encode/decode
- JPEG header generator/parser with enable/disable
- Four programmable quantization tables
- Single-clock Huffman coding and decoding
- Fully-programmable Huffman tables (two AC and two DC)
- Fully-programmable minimum coded unit (MCU)
- Concurrent input and output data stream interfaces
37.3 JPEG codec block functional description

37.3.1 General description

The block diagram of the JPEG codec is shown in Figure 277: JPEG codec block diagram.

![Figure 277. JPEG codec block diagram](MSv42133V1)

37.3.2 JPEG internal signals

Table 272 lists the JPEG internal signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jpeg_hclk</td>
<td>Digital input</td>
<td>JPEG kernel and register interface clock</td>
</tr>
<tr>
<td>jpeg_it</td>
<td>Digital output</td>
<td>JPEG global interrupt</td>
</tr>
<tr>
<td>jpeg_iftrg</td>
<td>Digital output</td>
<td>JPEG input FIFO threshold for MDMA</td>
</tr>
<tr>
<td>jpeg_ifnf_trg</td>
<td>Digital output</td>
<td>JPEG input FIFO not full for MDMA</td>
</tr>
<tr>
<td>jpeg_ofne_trg</td>
<td>Digital output</td>
<td>JPEG output FIFO threshold for MDMA</td>
</tr>
</tbody>
</table>
37.3.3 JPEG decoding procedure

The JPEG codec can decode a JPEG stream as defined in the ISO/IEC 10918-1 specification.

It can optionally parse the JPEG header and update accordingly the JPEG codec registers, the quantization tables and the Huffman tables.

The JPEG codec is configured in decode mode setting the DE bit (decode enable) of the JPEG_CONFR1 register.

The JPEG decode starts by setting the START bit of the JPEG_CONFR0 register.

The JPEG codec requests data for its input FIFO through generating one of:

- MDMA trigger
- Interrupts

Interrupt or MDMA trigger generation for input FIFO

Input FIFO can be managed using interrupts or MDMA triggers through two flags according to the FIFO state:

- Input FIFO not full flag: a 32-bit value can be written in.
- Input FIFO threshold flag: 8 words (32 bytes) can be written in.

The interrupt or MDMA trigger generation is independent of the START bit of the JPEG_CONFR0 register. The input FIFO flags are generated regardless of the state of the JPEG codec kernel.

Writes are ignored if the input FIFO is full.

At the end of the decoding process, extra bytes may remain in the input FIFO and/or an interrupt request / MDMA trigger may be pending. The FIFO can be flushed by setting the IFF bit (Input FIFO Flush) of the JPEG_CR register.

Prior to flushing the FIFO:

- The interrupts for the input FIFO must be disabled to prevent unwanted interrupt request upon flushing the FIFO.
- The MDMA channel must be stopped to prevent unwanted MDMA trigger.

The consequence of not flushing the FIFO at the end of the decoding process is that any remaining data is taken into the next JPEG decoding.

Header parsing

The header parsing can be activated setting the HDR bit of the JPEG_CONFR1 register.

The JPEG header parser supports all markers relevant to the JPEG baseline algorithm indicated in Annex B of the ISO/IEC 10918-1.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jpeg_ofne_trg</td>
<td>Digital output</td>
<td>JPEG output FIFO not empty for MDMA</td>
</tr>
<tr>
<td>jpeg_oec_trg</td>
<td>Digital output</td>
<td>JPEG end of conversion for MDMA</td>
</tr>
</tbody>
</table>
When parsing a supported marker, the JPEG header parser extracts the required parameters and stores them in shadow registers. At the end of the parsing the JPEG codec registers are updated.

If a DQT marker segment is located, quantization data associated with it is written into the quantization table memory.

If a DHT marker segment is located, the Huffman table data associated with it is converted into three different table formats (HuffMin, HuffBase and HuffSymb) and stored in their respective memories.

Once the parsing operation is completed, the HPDF (header parsing done flag) bit of the JPEG_SR register is set. An interrupt is generated if the EHPIE (end of header parsing interrupt enable) bit of the JPEG_CR register is set.

**JPEG decoding**

Once the JPEG header is parsed or JPEG codec registers and memories are properly programmed, the incoming data stream is decoded and the resulting MCUs are sent to the output FIFO.

When decoding two images successively, the START bit of the JPEG_CONFR0 register must be set again (even if already 1) after the header processing of the second image is completed.

**Interrupt or MDMA trigger generation for output FIFO**

The output FIFO can be managed using interrupts or MDMA triggers through two flags according to the FIFO state:

- Output FIFO not empty flag: a 32-bit value can be read out.
- Output FIFO Threshold flag: 8 words (32 bytes) can be read out.

Reads return 0 if the output FIFO is empty.

In case of abort of the JPEG codec operations by resetting the START bit of the JPEG_CONFR0 register, the output FIFO can be flushed. If the FIFO needs to be flushed, it shall be done by software setting the FF bit (FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO:
- The interrupts for the output FIFO must be disabled to prevent unwanted interrupt request upon flushing the FIFO.
- The MDMA channel must be stopped to prevent unwanted MDMA trigger.

The output FIFO must be flushed at the end of processing before any JPEG configuration change.

**37.3.4 JPEG encoding procedure**

The JPEG codec can encode a JPEG stream as defined in the ISO/IEC 10918-1 specification.

It can optionally generate the JPEG Header.

The JPEG codec is configured in encode mode resetting the DE bit (decode enable) of the JPEG_CONFR1 register.
The configuration used for encoding the JPEG must be loaded in the JPEG codec:
- JPEG codec configuration registers
- quantization tables
- Huffman tables

The JPEG codec is started setting the START bit of the JPEG_CONFR0 register.

Once the JPEG codec has been started, it request data for its input FIFO generating one of:
- MDMA trigger
- interrupts

**Interrupt or MDMA trigger generation for input FIFO**

Input FIFO can be managed using interrupts or MDMA triggers through two flags according to the FIFO state:
- Input FIFO not full flag: a 32-bit value can be written in.
- Input FIFO threshold flag: 8 words (32 bytes) can be written in.

The interrupt or MDMA trigger generation is independent of the START bit of the JPEG_CONFR0 register. The input FIFO flags are generated regardless of the state of the JPEG codec kernel.

Writes are ignored if the input FIFO is full.

At the end of the encoding process, extra bytes may remain in the input FIFO and/or an interrupt request / MDMA trigger may be pending. The FIFO can be flushed by setting the IFF bit (input FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO:
- The interrupts for the input FIFO must be disabled to prevent unwanted interrupt request upon flushing the FIFO.
- The MDMA channel must be stopped to prevent unwanted MDMA trigger.

The consequence of not flushing the FIFO at the end of the encoding process is that any remaining data is taken into the next JPEG encoding.

**JPEG encoding**

Once the JPEG header generated, the incoming MCUs are encoded and the resulting data stream sent to the output FIFO.

**Interrupt or MDMA trigger generation for output FIFO**

Output FIFO can be managed using interrupts or MDMA triggers through two flags according to the FIFO state:
- Output FIFO not empty flag: a 32-bit value can be read out.
- Output FIFO threshold flag: 8 words (32 bytes) can be read out.

Reads return 0 if the output FIFO is empty.

In case of abort of the JPEG codec operations by resetting the START bit of the JPEG_CONFR0 register, the output FIFO can be flushed. The FIFO can be flushed by setting the FF bit (FIFO flush) of the JPEG_CR register.
Prior to flushing the FIFO:
- The interrupts for the output FIFO must be disabled to prevent unwanted interrupt request upon flushing the FIFO.
- The MDMA channel must be stopped to prevent unwanted MDMA trigger.

The output FIFO must be flushed at the end of processing before any JPEG configuration change.

The EOCF bit (end of conversion flag) of the JPEG_SR register can only be cleared when the output FIFO is empty.

Clearing either of the HDR bit (header processing) of the JPEG_CONFR1 register and the JCEN bit (JPEG codec enable) of the JPEG_CR register is allowed only when the EOCF bit of the JPEG_SR register is cleared.

### 37.4 JPEG codec interrupts

An interrupt can be produced on the following events:
- input FIFO threshold reached
- input FIFO not full
- output FIFO threshold reached
- output FIFO not empty
- end of conversion
- header parsing done

Separate interrupt enable bits are available for flexibility.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input FIFO threshold reached</td>
<td>IFTF</td>
<td>IFTIE</td>
</tr>
<tr>
<td>Input FIFO not full</td>
<td>IFNFF</td>
<td>IFNFIE</td>
</tr>
<tr>
<td>Output FIFO threshold reached</td>
<td>OFTF</td>
<td>OFTIE</td>
</tr>
<tr>
<td>Output FIFO not empty</td>
<td>OFNEF</td>
<td>OFNEIE</td>
</tr>
<tr>
<td>End of conversion</td>
<td>EOCF</td>
<td>EOCIE</td>
</tr>
<tr>
<td>Header parsing done</td>
<td>HPDF</td>
<td>HPDIE</td>
</tr>
</tbody>
</table>
37.5 **JPEG codec registers**

37.5.1 **JPEG codec control register (JPEG_CONFR0)**

*Address offset: 0x0000*

*Reset value: 0x0000 0000*

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

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

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

**Bit 0** **START**: Start

This bit start or stop the encoding or decoding process.

0: Stop/abort
1: Start

Reads always return 0.

37.5.2 **JPEG codec configuration register 1 (JPEG_CONFR1)**

*Address offset: 0x0004*

*Reset value: 0x0000 0000*

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

**Bits 31:16** **YSIZE[15:0]**: Y Size

This field defines the number of lines in source image.

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

**Bit 8** **HDR**: Header processing

This bit enables the header processing (generation/parsing).

0: Disable
1: Enable

**Bits 7:6** **NS[1:0]**: Number of components for scan

This field defines the number of components minus 1 for scan header marker segment.
37.5.3 JPEG codec configuration register 2 (JPEG_CONFR2)

Address offset: 0x0008
Reset value: 0x0000 0000

Bits 5:4 COLSPACE[1:0]: Color space
This field defines the number of quantization tables minus 1 to insert in the output stream.
  00: Grayscale (1 quantization table)
  01: YUV (2 quantization tables)
  10: RGB (3 quantization tables)
  11: CMYK (4 quantization tables)

Bit 3 DE: Codec operation as coder or decoder
This bit selects the code or decode process
  0: Code
  1: Decode

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 NF[1:0]: Number of color components
This field defines the number of color components minus 1.
  00: Grayscale (1 color component)
  01: YUV or RGB (3 color components)
  11: CMYK (4 color components)

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 NMCU[25:0]: Number of MCUs
For encoding: this field defines the number of MCU units minus 1 to encode.
For decoding: this field indicates the number of complete MCU units minus 1 to be decoded (this field is updated after the JPEG header parsing). If the decoded image size has not a X or Y size multiple of 8 or 16 (depending on the sub-sampling process), the resulting incomplete or empty MCU must be added to this value to get the total number of MCUs generated.
### 37.5.4 JPEG codec configuration register 3 (JPEG_CONFR3)

Address offset: 0x000C  
Reset value: 0x0000 0000

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

Bits 31:16 $\text{XSIZE}[15:0]$: X size  
This field defines the number of pixels per line.  
Bits 15:0 Reserved, must be kept at reset value.

### 37.5.5 JPEG codec configuration register 4-7 (JPEG_CONFR4-7)

Address offset: 0x0010 + 0x4 * i, where “i” is image component from 0 to 3  
Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.  
Bits 31:12 $\text{HSF}[3:0]$: Horizontal sampling factor  
Horizontal sampling factor for component i.  
Bits 11:8 $\text{VSF}[3:0]$: Vertical sampling factor  
Vertical sampling factor for component i.  
Bits 7:4 $\text{NB}[3:0]$: Number of blocks  
Number of data units minus 1 that belong to a particular color in the MCU.
Bits 3:2 **QT[1:0]**: Quantization table
Selects quantization table used for component i.
- 00: Quantization table 0
- 01: Quantization table 1
- 10: Quantization table 2
- 11: Quantization table 3

Bit 1 **HA**: Huffman AC
Selects the Huffman table for encoding AC coefficients.
- 0: Huffman AC table 0
- 1: Huffman AC table 1

Bit 0 **HD**: Huffman DC
Selects the Huffman table for encoding DC coefficients.
- 0: Huffman DC table 0
- 1: Huffman DC table 1

### 37.5.6 JPEG control register (JPEG_CR)

**Address offset**: 0x0030

**Reset value**: 0x0000 0000

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

Bits 31:15 Reserved, must be kept at reset value.

**Bit 14 OFF**: Output FIFO flush
This bit flushes the output FIFO.
- 0: No effect
- 1: Output FIFO is flushed
Reads always return 0.

**Bit 13 IFF**: Input FIFO flush
This bit flushes the input FIFO.
- 0: No effect
- 1: Input FIFO is flushed
Reads always return 0.

**Bits 12:7** Reserved, must be kept at reset value.

**Bit 6 HPDIE**: Header parsing done interrupt enable
This bit enables interrupt generation upon the completion of the header parsing operation.
- 0: Disabled
- 1: Enabled
Bit 5  **EOCIE**: End of conversion interrupt enable  
This bit enables interrupt generation at the end of conversion.  
0: Disabled  
1: Enabled  

Bit 4  **OFNIE**: Output FIFO not empty interrupt enable  
This bit enables interrupt generation when the output FIFO is not empty.  
0: Disabled  
1: Enabled  

Bit 3  **OFTIE**: Output FIFO threshold interrupt enable  
This bit enables interrupt generation when the output FIFO reaches a threshold.  
0: Disabled  
1: Enabled  

Bit 2  **IFNFIE**: Input FIFO not full interrupt enable  
This bit enables interrupt generation when the input FIFO is not empty.  
0: Disabled  
1: Enabled  

Bit 1  **IFTIE**: Input FIFO threshold interrupt enable  
This bit enables interrupt generation when the input FIFO reaches a threshold.  
0: Disabled  
1: Enabled  

Bit 0  **JCEN**: JPEG core enable  
This bit enables the JPEG codec core.  
0: Disabled (internal registers are reset).  
1: Enabled (internal registers are accessible).  

### 37.5.7 JPEG status register (JPEG_SR)

Address offset: 0x0034  
Reset value: 0x0000 0006

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

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

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>COF</td>
<td>HPDF</td>
<td>EOCF</td>
<td>OFNFT</td>
<td>OFTF</td>
<td>IFNF</td>
<td>IFTF</td>
<td>RES.</td>
<td>RES.</td>
<td>RES.</td>
<td>RES.</td>
<td>RES.</td>
<td>RES.</td>
<td>RES.</td>
<td>RES.</td>
<td>RES.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
37.5.8 JPEG clear flag register (JPEG_CFR)

Address offset: 0x0038

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ChPDF</td>
<td>CEOCF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rt_w1</td>
<td>rt_w1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:7  Reserved, must be kept at reset value.

Bit 6  **CHPDF**: Clear header parsing done flag
Writing 1 clears the HPDF bit of the JPEG_SR register.
   0: No effect
   1: Clear

Bit 5  **CEOCF**: Clear end of conversion flag
Writing 1 clears the ECF bit of the JPEG_SR register.
   0: No effect
   1: Clear

Bits 4:0  Reserved, must be kept at reset value.

### 37.5.9  JPEG data input register (JPEG_DIR)

**Address offset**: 0x0040

**Reset value**: 0x0000 0000

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

Bits 31:0  **DATAIN[31:0]**: Data input FIFO
Input FIFO data register.

### 37.5.10  JPEG data output register (JPEG_DOR)

**Address offset**: 0x0044

**Reset value**: 0x0000 0000

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

Bits 31:0  **DATAOUT[31:0]**: Data output FIFO
Output FIFO data register.
### 37.5.11 JPEG quantization memory x (JPEG_QMEMx\[y\])

Address offset: 0x0050 + x*64 + 4*y, x = 0..3 and y = 0..15

Reset value: 0xXXXX XXXX

4 quantization tables as specified by ISO documentation.

For decoding with header parsing, no quantization table programming is required, the coefficients are directly written in the quantization memories by header parser.

For decoding without header parsing or for encoding, the quantization table must be written by software in zig zag order.

| Bits 31:24 | QCOEF<4*y+3>[7:0] | Quantization coefficient 4*y+3 | 8-bit quantization coefficient. |
| Bits 23:16 | QCOEF<4*y+2>[7:0] | Quantization coefficient 4*y+2 | 8-bit quantization coefficient. |
| Bits 15:8  | QCOEF<4*y+1>[7:0] | Quantization coefficient 4*y+1 | 8-bit quantization coefficient. |
| Bits 7:0   | QCOEF<4*y>[7:0]  | Quantization coefficient 4*y   | 8-bit quantization coefficient. |

| DATA0: Min AC0 value |
| DATA1: Min DC0 value |
| DATA2: Min AC1 value |
| DATA3: Min DC1 value |

---

### 37.5.12 JPEG Huffman min (JPEG_HUFFMIN[4*y+x])

Address offset: 0x0150 + 16*y + 4*x, x = 0..3 and y = 0..3

Reset value: 0xXXXX XXXX

This memory stores the minimum Huffman values used internally by the JPEG decoder. The memory content is written by hardware during the header parsing.
37.5.13 JPEG Huffman base (JPEG_HUFFBASE[y])

Address offset: 0x0190 + 4*y, y = 0..31
Reset value: 0xXXXX XXXX

This memory stores the base Huffman values used internally by the JPEG decoder. The memory content is written by hardware during the header parsing:

- DATA0 to DATA15: Base AC0 value
- DATA16 to DATA31: Base DC0 value
- DATA32 to DATA47: Base AC1 value
- DATA48 to DATA63: Base DC1 value

Bits 31:0 \( \text{DATA}_y[99:0] \): Minimum Huffman value
100-bit minimum Huffman value used internally by the JPEG decoder.

Bits 24:16 \( \text{DATA}_y[8:0] \): DATA \( 2^y+1 \)
Base Huffman value.

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

Bits 8:0 \( \text{DATA}_y[8:0] \): DATA \( 2^y \)
Base Huffman value.
37.5.14 JPEG Huffman symbol (JPEG_HUFFSYMB[y])

Address offset: 0x0210 + 4\*y, y = 0..83
Reset value: 0xXXXX XXXX

This memory stores the Huffman symbols used internally by the JPEG decoder. The memory content is written by hardware during the header parsing:
- DATA0 to DATA161: AC0 symbols
- DATA162 to DATA173: DC0 and DC1 symbols
- DATA174 to DATA335: AC1 symbols

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0210 + 4*y</td>
<td>Huffman symbol</td>
</tr>
</tbody>
</table>

37.5.15 JPEG DHT memory (JPEG_DHTMEM[y])

Address offset: 0x0360 + 4\*y, y = 0..102
Reset value: 0xXXXX XXXX

For encoding process with header generation, this memory stores the DHT marker segment AC and DC Huffman tables in the ISO/IEC specification format:
- DATA0 to DATA27: DC Huffman table0
- DATA28 to DATA233: AC Huffman table0
- DATA234 to DATA411: DC Huffman table1
- DATA412 to DATA687: AC Huffman table1

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0360 + 4*y</td>
<td>Huffman symbol</td>
</tr>
</tbody>
</table>
37.5.16 JPEG Huffman encoder ACx (JPEG_HUFFENC_ACx[y])

Address offset: 0x0500 + 220*x + 4*y, x=0..1, y = 0..87
Reset value: 0xXXXX XXXX

This memory defines the Huffman codes used during the encoding process of AC components.

<table>
<thead>
<tr>
<th>Bits 31:24</th>
<th>DATA&lt;4*y+3&gt;[7:0]</th>
<th>Huffman table data 4*y+3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Huffman table data for DHT marker segment generation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 23:16</th>
<th>DATA&lt;4*y+2&gt;[7:0]</th>
<th>Huffman table data 4*y+2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Huffman table data for DHT marker segment generation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:8</th>
<th>DATA&lt;4*y+1&gt;[7:0]</th>
<th>Huffman table data 4*y+1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Huffman table data for DHT marker segment generation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 7:0</th>
<th>DATA&lt;4*y&gt;[7:0]</th>
<th>Huffman table data 4*y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Huffman table data for DHT marker segment generation.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bits 31:28</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bits 27:24 HLEN&lt;2<em>y+1&gt;[3:0]: Huffman length 2</em>y+1</td>
</tr>
<tr>
<td></td>
<td>Number of bits in the Huffman code HCODE&lt;2*y+1&gt; minus 1.</td>
</tr>
<tr>
<td></td>
<td>Bits 23:16 HCODE&lt;2<em>y+1&gt;[7:0]: Huffman code 2</em>y+1</td>
</tr>
<tr>
<td></td>
<td>8 least significant bits of the Huffman code.</td>
</tr>
<tr>
<td></td>
<td>If the Huffman code is less than 8 bits long, the unused bits must be 0.</td>
</tr>
<tr>
<td></td>
<td>Bits 15:12 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td></td>
<td>Bits 11:8 HLEN&lt;2<em>y&gt;[3:0]: Huffman length 2</em>y</td>
</tr>
<tr>
<td></td>
<td>Number of bits in the Huffman code HCODE&lt;2*y&gt; minus 1.</td>
</tr>
<tr>
<td></td>
<td>Bits 7:0 HCODE&lt;2<em>y&gt;[7:0]: Huffman code 2</em>y</td>
</tr>
<tr>
<td></td>
<td>8 least significant bits of the Huffman code.</td>
</tr>
<tr>
<td></td>
<td>If the Huffman code is less than 8 bits long, the unused bits must be 0.</td>
</tr>
</tbody>
</table>

37.5.17 JPEG Huffman encoder DCx (JPEG_HUFFENC_DCx[y])

Address offset: 0x07C0 + 220*x + 4*y, x=0..1, y = 0..7
Reset value: 0xXXXX XXXX

This memory defines the Huffman codes used during the encoding process of DC components.
Bits 31:28  Reserved, must be kept at reset value.

Bits 27:24  **HLEN<2*y+1>[3:0]**: Huffman length $2^y+1$
            Number of bits in the Huffman code HCODE<2*y+1> minus 1.

Bits 23:16  **HCODE<2*y+1>[7:0]**: Huffman code $2^y+1$
            8 least significant bits of the Huffman code.
            If the Huffman code is less than 8 bits long, the unused bits must be 0.

Bits 15:12  Reserved, must be kept at reset value.

Bits 11:8  **HLEN<2*y>[3:0]**: Huffman length $2^y$
            Number of bits in the Huffman code HCODE<2*y> minus 1.

Bits 7:0  **HCODE<2*y>[7:0]**: Huffman code $2^y$
            8 least significant bits of the Huffman code.
            If the Huffman code is less than 8 bits long, the unused bits must be 0.
## 37.5.18 JPEG codec register map

The following table summarizes the JPEG codec registers. Refer to the register boundary addresses table for the JPEG codec register base address.

### Table 274. JPEG codec register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0030</td>
<td>JPEG_CR</td>
<td>OFF</td>
<td>IFF</td>
<td>HD</td>
<td>EODC</td>
<td>ONSF</td>
<td>OFS</td>
<td>OFN</td>
<td>OTE</td>
<td>IFNE</td>
<td>IPE</td>
<td>JCEN</td>
</tr>
<tr>
<td>0x0034</td>
<td>JPEG_SR</td>
<td>OFF</td>
<td>IFF</td>
<td>HD</td>
<td>EODC</td>
<td>ONSF</td>
<td>OFS</td>
<td>OFN</td>
<td>OTE</td>
<td>IFNE</td>
<td>IPE</td>
<td>JCEN</td>
</tr>
<tr>
<td>0x0038</td>
<td>JPEG_CFR</td>
<td>OFF</td>
<td>IFF</td>
<td>HD</td>
<td>EODC</td>
<td>ONSF</td>
<td>OFS</td>
<td>OFN</td>
<td>OTE</td>
<td>IFNE</td>
<td>IPE</td>
<td>JCEN</td>
</tr>
</tbody>
</table>

Reset values:
- JPEG_CONFR0: 0
- JPEG_CONFR1: 0
- JPEG_CONFR2: 0
- JPEG_CONFR3: 0
- JPEG_CONFR4: 0
- JPEG_CONFR5: 0
- JPEG_CONFR6: 0
- JPEG_CONFR7: 0
- JPEG_CR: 0
- JPEG_SR: 0
- JPEG_CFR: 0
- JPEG_DIR: 0
Table 274. JPEG codec register map and reset values (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0044</td>
<td>JPEG_DOR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0048-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0050-</td>
<td>JPEG_QMEM0</td>
<td>QCOEF&lt;y*4+3&gt;</td>
<td>QCOEF&lt;y*4+2&gt;</td>
<td>QCOEF&lt;y*4+1&gt;</td>
<td>QCOEF&lt;y*4&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008C-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0090-</td>
<td>JPEG_QMEM1</td>
<td>QCOEF&lt;y*4+3&gt;</td>
<td>QCOEF&lt;y*4+2&gt;</td>
<td>QCOEF&lt;y*4+1&gt;</td>
<td>QCOEF&lt;y*4&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00CC-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00D0-</td>
<td>JPEG_QMEM2</td>
<td>QCOEF&lt;y*4+3&gt;</td>
<td>QCOEF&lt;y*4+2&gt;</td>
<td>QCOEF&lt;y*4+1&gt;</td>
<td>QCOEF&lt;y*4&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010F-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0110-</td>
<td>JPEG_QMEM3</td>
<td>QCOEF&lt;y*4+3&gt;</td>
<td>QCOEF&lt;y*4+2&gt;</td>
<td>QCOEF&lt;y*4+1&gt;</td>
<td>QCOEF&lt;y*4&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x014C-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0150-</td>
<td>JPEG_HUFFMIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x018C-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0190-</td>
<td>JPEG_HUFFBASE</td>
<td>DATA&lt;y&gt;8:0</td>
<td>DATA&lt;y&gt;31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01B4-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0210-</td>
<td>JPEG_HUFFSYM</td>
<td>DATA&lt;y&gt;4+3&gt;</td>
<td>DATA&lt;y&gt;4+2&gt;</td>
<td>DATA&lt;y&gt;4+1&gt;</td>
<td>DATA&lt;y&gt;4&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x023C-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0360-</td>
<td>JPEG_DHTMEM</td>
<td>DATA&lt;y&gt;4+3&gt;</td>
<td>DATA&lt;y&gt;4+2&gt;</td>
<td>DATA&lt;y&gt;4+1&gt;</td>
<td>DATA&lt;y&gt;4&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04F8-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0500-</td>
<td>JPEG_HUFFENC</td>
<td>HLEN&lt;2&gt;y+1</td>
<td>HCODE&lt;2&gt;y+1</td>
<td>HLEN&lt;2&gt;y+1</td>
<td>HCODE&lt;2&gt;y+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x05C4-</td>
<td>_AC0</td>
<td>[3:0]</td>
<td>[3:0]</td>
<td>[3:0]</td>
<td>[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0660-</td>
<td>JPEG_HUFFENC</td>
<td>HLEN&lt;2&gt;y+1</td>
<td>HCODE&lt;2&gt;y+1</td>
<td>HLEN&lt;2&gt;y+1</td>
<td>HCODE&lt;2&gt;y+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x06C4-</td>
<td>_AC1</td>
<td>[3:0]</td>
<td>[3:0]</td>
<td>[3:0]</td>
<td>[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x07CC-</td>
<td>JPEG_HUFFENC</td>
<td>HLEN&lt;2&gt;y+1</td>
<td>HCODE&lt;2&gt;y+1</td>
<td>HLEN&lt;2&gt;y+1</td>
<td>HCODE&lt;2&gt;y+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0820-</td>
<td>_DC0</td>
<td>[3:0]</td>
<td>[3:0]</td>
<td>[3:0]</td>
<td>[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0880-</td>
<td>JPEG_HUFFENC</td>
<td>HLEN&lt;2&gt;y+1</td>
<td>HCODE&lt;2&gt;y+1</td>
<td>HLEN&lt;2&gt;y+1</td>
<td>HCODE&lt;2&gt;y+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08E4-</td>
<td>_DC1</td>
<td>[3:0]</td>
<td>[3:0]</td>
<td>[3:0]</td>
<td>[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
38 True random number generator (RNG)

38.1 Introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG is a NIST SP 800-90B compliant entropy source that can be used to construct a non-deterministic random bit generator (NDRBG).

The RNG true random number generator has been pre-certified NIST SP800-90B. It has also been tested using German BSI statistical tests of AIS-31 (T0 to T8).

38.2 RNG main features

- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by a NIST SP800-90B approved conditioning stage.
- It can be used as entropy source to construct a non-deterministic random bit generator (NDRBG).
- It produces four 32-bit random samples every 412 AHB clock cycles if $f_{\text{AHB}} < 77$ MHz (256 RNG clock cycles otherwise).
- It embeds start-up and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management.
- It can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- It has an AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).
38.3 RNG functional description

38.3.1 RNG block diagram

Figure 278 shows the RNG block diagram.

38.3.2 RNG internal signals

Table 275 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rng_it</td>
<td>Digital output</td>
<td>RNG global interrupt request</td>
</tr>
<tr>
<td>rng_hclk</td>
<td>Digital input</td>
<td>AHB clock</td>
</tr>
<tr>
<td>rng_clk</td>
<td>Digital input</td>
<td>RNG dedicated clock, asynchronous to rng_hclk</td>
</tr>
</tbody>
</table>
38.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary RNG integrates all the required NIST components depicted on Figure 279. Those components are an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

Figure 279. NIST SP800-90B entropy source model

The components pictured above are detailed hereafter:

**Noise source**

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x6), each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in Section 38.3.8: RNG low-power usage.
- The XORing of the 6 noise sources into a single analog output.
- A sampling stage of this output clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (rng_hclk), with a possibility for the software to decrease the sampling frequency by using the integrated divider.
Note: In Section 38.6: RNG entropy source validation recommended RNG clock frequencies and associated divider value are given.

**Post processing**

In NIST configuration no post-processing is applied to sampled noise source. In non-NIST configuration B (as defined in Section 38.6.2) a normalization debiasing is applied, i.e. half of the bits are taken from the sampled noise source, half of the bits are taken from inverted sampled noise source.

**Conditioning**

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit). The NIST SP800-90B target is full entropy on the output (i.e. 128-bit).

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in Section 38.5: RNG processing time.

**Output buffer**

A data output buffer can store up to four 32-bit words that have been output from the conditioning component. When four words have been read from the output FIFO through the RNG DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register after a number of clock cycles specified in Section 38.5: RNG processing time.

Whenever a random number is available through the RNG DR register the DRDY flag transitions from “0” to “1”. This flag remains high until output buffer becomes empty after reading four words from the RNG DR register.

Note: When interrupts are enabled an interrupt is generated when this data ready flag transitions from “0” to “1”. Interrupt is then cleared automatically by the RNG as explained above.
Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the value recommended for register RNG_HTCR (in Section 38.6.2).

1. Start-up health tests, performed after reset and before the first use of the RNG as entropy source
   - Adaptive proportion test running on one 1024 bit windows: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 691 times.
   - Known-answer tests, to verify the conditioning stage.
   - Repetition count test, flagging an error when the noise source has provided more than 40 consecutive bits at a constant value (“0” or “1”)

2. Continuous health tests, running indefinitely on the outputs of the noise source
   - Repetition count test, similar to the one running in start-up tests
   - Adaptive proportion test running on 1024 consecutive samples, like during start-up health tests.

3. Vendor specific continuous tests
   - Transition count test, flagging an error when the noise source has delivered more than 32 consecutive occurrence of two bits patterns (“01” or “10”).
   - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than AHB clock cycle divided by 32.

4. On-demand test of digitized noise source (raw data)
   - Supported by restarting the entropy source and re-running the startup tests (see software reset sequence in Section 38.3.4: RNG initialization). Other kinds of on-demand testing (software based) are not supported.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in Section 38.3.7: Error management.

Note: An interrupt can be generated when an error is detected.

Above health test thresholds are modified by changing value in RNG_HTCR register. See Section 38.6: RNG entropy source validation for details.

38.3.4 RNG initialization

The RNG simplified state machine is pictured on Figure 280
After enabling the RNG (RNGEN=1 in RNG_CR) the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 cycles of RNG clock cycles (before divider) before starting to sample analog output and filling 128-bit conditioning shift register.

2. The conditioning hardware initializes, automatically triggering start-up behavior test on the raw data samples and known-answer tests.

3. When start-up health tests are completed. During this time three 128-bit noise source samples are used.

4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.

5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in Section 38.5: RNG processing time.

Figure 280 also highlights a possible software reset sequence, implemented by:

a) Writing bits RNGEN=0 and CONDRST=1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.

b) Then writing RNGEN=1 and CONDRST=0 in the RNG_CR register.

c) Wait for random number to be ready, after initialization completes.
Note: When RNG peripheral is reset through RCC (hardware reset) the RNG configuration for optimal randomness is lost in RNG registers. Software reset with CONFIGLOCK set preserves the RNG configuration.

38.3.5 RNG operation

Normal operations

To run the RNG using interrupts the following steps are recommended:

1. Consult the Section 38.6: RNG entropy source validation on page 1318 and verify if a specific RNG configuration is required for your application.
   - If it is the case, write in the RNG_CR register the bit CONDRST="1" together with the correct RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST="0", the interrupt enable bit IE="1" and the RNG enable bit RNGEN="1".
   - If it is not the case perform a write to the RNG_CR register with the interrupt enable bit IE="1" and the RNG enable bit RNGEN="1".

2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits should be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data is available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of above conditions are false, the RNG_DR register must not be read. If an error occurred error recovery sequence described in Section 38.3.7 shall be used.

To run the RNG in polling mode following steps are recommended:

1. Consult the Section 38.6: RNG entropy source validation on page 1318 and verify if a specific RNG configuration is required for your application.
   - If it is the case write in the RNG_CR register the bit CONDRST="1" together with the correction RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST="0" and the RNG enable bit RNGEN="1".
   - If it is not the case only enable the RNG by setting the RNGEN bit to "1" in the RNG_CR register.

2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits should be set to 0)
   - A random number is ready (the DRDY bit should be set to 1)

3. If above conditions are true read the content of the RNG_DR register up to four consecutive times. If valid data is available in the conditioning output buffer four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of above conditions are false, the RNG_DR register must not be read. If an error occurred error recovery sequence described in Section 38.3.7 shall be used.

Note: When data is not ready (DRDY="0") RNG_DR returns zero.
It is recommended to always verify that RNG_DR is different from zero. Because when it is
the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).

If the random number generation period is a concern to the application and if NIST compliance is not required it is possible to select a faster RNG configuration by using the RNG configuration “B”, described in Section 38.6: RNG entropy source validation. The gain in random number generation speed is summarized in Section 38.5: RNG processing time.

**Low-power operations**

If the power consumption is a concern to the application, low-power strategies can be used, as described in Section 38.3.8: RNG low-power usage.

**Software post-processing**

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals.

Built-in health check functions are described in Section 38.3.3: Random number generation.

### 38.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in Section 38.6: RNG entropy source validation.

**Note:** When the CED bit in the RNG_CR register is set to “0”, the RNG clock frequency before internal divider should be higher than AHB clock frequency divided by 32, otherwise the clock checker always flags a clock error (CECS=1 in the RNG_SR register).

See Section 38.3.1: RNG block diagram for details (AHB and RNG clock domains).

### 38.3.7 Error management

In parallel to random number generation an health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

**Clock error detection**

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to “1” both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application should check that the RNG clock is configured correctly (see Section 38.3.6: RNG clocking) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when clocking condition is normal.

**Note:** The clock error has no impact on generated random numbers, i.e. application can still read RNG_DR register.

CEIS is set only when CECS is set to “1” by RNG.
Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to “1” both SEIS and SECS bits to indicate that a seed error occurred. If a value is available in the RNG_DR register, it must not be used as it may not have enough entropy.

The following sequence shall be used to fully recover from a seed error:
1. Software reset by writing CONDRST at “1” and at “0” (see bitfield description for details).
2. Wait for CONDRST to be cleared in the RNG_CR register, then confirm that SEIS is cleared in the RNG_SR register.
3. Wait for SECS to be cleared by RNG. The random number generation is now back to normal.

38.3.8 RNG low-power usage

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to “1” by setting the RNGEN bit to “0” in the RNG_CR register. As the post-processing logic and the output buffer remain operational while RNGEN=’0’ following features are available to software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG_DR register. If it is not the case RNG must be re-enabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode) and a complete conditioning round has been done. Four new random words are then available only if the expected number of conditioning round has been reached (two if NISTC=0). The overall time can be found in Section 38.5: RNG processing time on page 1317.

When disabling the RNG the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (i.e. well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when RNGEN bit is set to “1”, unless the application resets the conditioning logic using CONDRST bit in the RNG_CR register.

38.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see Section 38.3.7: Error management
- Clock error, see Section 38.3.7: Error management

Dedicated interrupt enable control bits are available as shown in Table 276.
The user can enable or disable the above interrupt sources individually by changing the mask bits or the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

### 38.5 RNG processing time

In NIST compliant configuration, the time between two sets of four 32-bit data is either:

- 412 AHB cycles if $f_{\text{AHB}} < f_{\text{threshold}}$ (conditioning stage is limiting), or
- 256 RNG cycles $f_{\text{AHB}} \geq f_{\text{threshold}}$ (noise source stage is limiting).

With $f_{\text{threshold}} = 1.6 \times f_{\text{AHB}}$, e.g. 77 MHz if $f_{\text{RNG}} = 48$ MHz.

Note: When CLKDIV is different from zero, $f_{\text{RNG}}$ must take into account the internal divider ratio.
### 38.6 RNG entropy source validation

#### 38.6.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite. The results can be provided on demand or the customer can reproduce the tests.

#### 38.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock `rng_clk= 48 MHz`
- RNG configurations described in *Table 277*. Note that only configuration A can be certified NIST SP800-90B.

#### 38.6.3 Data collection

In order to run statistical tests it is required to collect samples from the entropy source at raw data level as well as at the output of the entropy source. For details on data collection and the running of statistical test suites refer to “STM32 microcontrollers random number generation validation using NIST statistical test suite” application note (AN4230) available from [www.st.com](http://www.st.com).

Contact STMicroelectronics if above samples need to be retrieved for your product.

---

**Table 277. RNG configurations**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0x0F</td>
<td>0x0</td>
<td>0x0</td>
<td>0xD</td>
<td>0</td>
<td>2</td>
<td>0x0000</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0x18</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0</td>
<td>2</td>
<td>A2B3</td>
</tr>
</tbody>
</table>

1. When writing this register magic number 0x17590ABC must be written immediately before the indicated value
38.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

38.7.1 RNG control register (RNG_CR)

Address offset: 0x000
Reset value: 0x0080 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>CONFIGLOCK: RNG Config Lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>Writes to the RNG_CR configuration bits [29:4] are allowed.</td>
</tr>
<tr>
<td>1:</td>
<td>Writes to the RNG_CR configuration bits [29:4] are ignored until the next RNG reset.</td>
</tr>
<tr>
<td>This bitfield is set once, i.e. if this bit is set it can only be reset to 0 if RNG is reset.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>CONDRST: Conditioning soft reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_CR and RNG_NSCR are not changed by CONDRST.</td>
<td></td>
</tr>
<tr>
<td>This bit must be set to 1 in the same access that set any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.</td>
<td></td>
</tr>
<tr>
<td>When CONDRST is set to 0 by software its value goes to 0 when the reset process is done. It takes about 2 AHB clock cycles + 2 RNG clock cycles.</td>
<td></td>
</tr>
</tbody>
</table>

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:20 RNG_CONFIG1[5:0]: RNG configuration 1
Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in Section 38.6: RNG entropy source validation.
Writing any bit of RNG_CONFIG1 is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK=1.

Bits 19:16 CLKDIV[3:0]: Clock divider factor
This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN=0).
0x0: internal RNG clock after divider is similar to incoming RNG clock.
0x1: two RNG clock cycles per internal RNG clock.
...
0xF: 2\(^{15}\) RNG clock cycles per internal clock, e.g. an incoming 48MHz RNG clock becomes a 1.5 kHz internal RNG clock.
Writing these bits is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK=1.

Bits 15:13 RNG_CONFIG2[2:0]: RNG configuration 2
Reserved to the RNG configuration (bitfield 2). Refer to RNG_CONFIG1 bitfield for details.
Bit 12 **NISTC**: Non NIST compliant

0: Hardware default values for NIST compliant RNG. In this configuration per 128-bit output two conditioning loops are performed and 256 bits of noise source are used.

1: Custom values for NIST compliant RNG. See Section 38.6: RNG entropy source validation for proposed configuration.

Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK=1.

Bits 11:8 **RNG_CONFIG3[3:0]**: RNG configuration 3

Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details.

If NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CED**: Clock error detection

0: Clock error detection is enable

1: Clock error detection is disable

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, i.e. to enable or disable CED the RNG must be disabled.

Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK=1.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE**: Interrupt Enable

0: RNG Interrupt is disabled

1: RNG Interrupt is enabled. An interrupt is pending as soon as DRDY='1', SEIS='1' or CEIS=1 in the RNG_SR register.

Bit 2 **RNGEN**: True random number generator enable

0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.

1: True random number generator is enabled.

Bits 1:0 Reserved, must be kept at reset value.
38.7.2 RNG status register (RNG_SR)

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rc_w0</td>
<td>rc_w0</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 SEIS: Seed error interrupt status
This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used).
Writing 1 has no effect.
0: No faulty sequence detected
1: At least one faulty sequence has been detected. See SECS bit description for details.
An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 CEIS: Clock error interrupt status
This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.
0: The RNG clock is correct (fRNGCLK > fHCLK/32)
1: The RNG clock before internal divider has been detected too slow (fRNGCLK < fHCLK/32)
An interrupt is pending if IE = 1 in the RNG_CR register.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 SECS: Seed error current status
0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.
1: At least one of the following faulty sequence has been detected:
   – Run-time repetition count test failed (noise source has provided more than 24 consecutive bits at a constant value 0" or 1", or more than 32 consecutive occurrence of two bits patterns 01" or 10")
   – Start-up or continuous adaptive proportion test on noise source failed.
   – Start-up post-processing/conditioning sanity check failed.

Bit 1 CECS: Clock error current status
0: The RNG clock is correct (fRNGCLK > fHCLK/32). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.
1: The RNG clock is too slow (fRNGCLK < fHCLK/32).
Note: CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 DRDY: Data Ready
0: The RNG_DR register is not yet valid, no random data is available.
1: The RNG_DR register contains valid random data.
Once the output buffer becomes empty (after reading the RNG_DR register), this bit returns to 0 until a new random value is generated.
Note: The DRDY bit can rise when the peripheral is disabled (RNGEN=0 in the RNG_CR register).
If IE=1 in the RNG_CR register, an interrupt is generated when DRDY=1.
38.7.3 RNG data register (RNG_DR)

Address offset: 0x008
Reset value: 0x0000 0000

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when DRDY=1 and value is not 0x0, even if RNGEN=0.

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

RNDATA[15:0]:

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

Bits 31:0 RNDATA[31:0]: Random data

32-bit random data which are valid when DRDY=1. When DRDY=0 RNDATA value is zero. It is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).

38.7.4 RNG health test control register (RNG_HTCR)

Address offset: 0x010
Reset value: 0x000C AA74

Writing in RNG_HTCR is taken into account only if CONDRST bit is set, and CONFIGLOCK bit is cleared in RNG_CR. Writing to this register is ignored if CONFIGLOCK=1.

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

HTCFG[15:0]:

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

Bits 31:0 HTCFG[31:0]: health test configuration

This configuration is used by RNG to configure the health tests. See Section 38.6: RNG entropy source validation for the recommended value.

Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.

When reading or writing this register magic number; 0x17590ABC must be written immediately before to RNG_HTCR register.
38.7.5 RNG register map

Table 278 gives the RNG register map and reset values.

| Offset | Register name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x000  | RNG_CR        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x004  | RNG_SR        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x008  | RNG_DR        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x010  | RNG_HTCR      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 for the register boundary addresses.
39 Cryptographic processor (CRYP)

39.1 Introduction

The cryptographic processor (CRYP) can be used both to encrypt and decrypt data using the DES, Triple-DES or AES algorithms. It is a fully compliant implementation of the following standards:

- The data encryption standard (DES) and Triple-DES (TDES) as defined by Federal Information Processing Standards Publication (FIPS PUB 46-3, Oct 1999), and the American National Standards Institute (ANSI X9.52)
- The advanced encryption standard (AES) as defined by Federal Information Processing Standards Publication (FIPS PUB 197, Nov 2001)

Multiple key sizes and chaining modes are supported:

- DES/TDES chaining modes ECB and CBC, supporting standard 56-bit keys with 8-bit parity per key
- AES chaining modes ECB, CBC, CTR, GCM, GMAC, CCM for key sizes of 128, 192 or 256 bits

The CRYP is a 32-bit AHB peripheral. It supports DMA transfers for incoming and outgoing data (two DMA channels are required). The peripheral also includes input and output FIFOs (each 8 words deep) for better performance.

The CRYP peripheral provides hardware acceleration to AES and DES cryptographic algorithms packaged in STM32 cryptographic library.

39.2 CRYP main features

- Compliant implementation of the following standards:
  - NIST FIPS publication 46-3, Data Encryption Standard (DES)
  - ANSI X9.52, Triple Data Encryption Algorithm Modes of Operation
  - NIST FIPS publication 197, Advanced Encryption Standard (AES)
- AES symmetric block cipher implementation
  - 128-bit data block processing
  - Support for 128-, 192- and 256-bit cipher key lengths
  - Encryption and decryption with multiple chaining modes: Electronic Code Book (ECB), Cipher Block Chaining (CBC), Counter mode (CTR), Galois Counter Mode (GCM), Galois Message Authentication Code mode (GMAC) and Counter with CBC-MAC (CCM).
  - 14 (respectively 18) clock cycles for processing one 128-bit block of data with a 128-bit (respectively 256-bit) key in AES-ECB mode
  - Integrated key scheduler with its key derivation stage (ECB or CBC decryption only)
- DES/TDES encryption/decryption implementation
  - 64-bit data block processing
  - Support for 64-, 128- and 192-bit cipher key lengths (including parity)
  - Encryption and decryption with support of ECB and CBC chaining modes
- Direct implementation of simple DES algorithms (a single key K1 is used)
- 16 (respectively 48) clock cycles for processing one 64-bit block of data in DES (respectively TDES) ECB mode
- Software implementation of ciphertext stealing

- Features common to DES/TDES and AES
  - AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
  - 256-bit register for storing the cryptographic key (8x 32-bit registers)
  - 128-bit registers for storing initialization vectors (4x 32-bit)
  - 1x32-bit INPUT buffer associated with an internal IN FIFO of eight 32-bit words, corresponding to four incoming DES blocks or two AES blocks
  - 1x32-bit OUTPUT buffer associated with an internal OUT FIFO of eight 32-bit words, corresponding to four processed DES blocks or two AES blocks
  - Automatic data flow control supporting direct memory access (DMA) using two channels (one for incoming data, one for processed data). Single and burst transfers are supported.
  - Data swapping logic to support 1-, 8-, 16- or 32-bit data
  - Possibility for software to suspend a message if the cryptographic processor needs to process another message with higher priority (suspend/resume operation)
39.3 CRYP functional description

39.3.1 CRYP block diagram

*Figure 281* shows the block diagram of the cryptographic processor.

*Figure 281. CRYP block diagram*
39.3.2 CRYP internal signals

Table 279 provides a list of useful-to-know internal signals available at cryptographic processor level and not at STM32 product level (on pads).

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cryp_hclk</td>
<td>digital input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>cryp_it</td>
<td>digital output</td>
<td>Cryptographic processor global interrupt request</td>
</tr>
<tr>
<td>cryp_in_dma</td>
<td>digital input/output</td>
<td>IN FIFO DMA burst request/ acknowledge</td>
</tr>
<tr>
<td>cryp_out_dma</td>
<td>digital input/output</td>
<td>OUT FIFO DMA burst request/ acknowledge (with single request for DES)</td>
</tr>
</tbody>
</table>

39.3.3 CRYP DES/TDES cryptographic core

Overview

The DES/Triple-DES cryptographic core consists of three components:

- The DES Algorithm (DEA core)
- Multiple keys (one for the DES algorithm, one to three for the TDES algorithm)
- The initialization vector, which is used only in CBC mode

The DES/Triple-DES cryptographic core provides two operating modes:

- **ALGODIR=0**: Plaintext encryption using the key stored in the CRYP_Kx registers.
- **ALGODIR=1**: Ciphertext decryption using the key stored in the CRYP_Kx registers.

The operating mode is selected by programming the ALGODIR bit in the CRYP_CR register.

Typical data processing

Typical usage of the cryptographic processor in DES modes can be found in Section 39.3.10: CRYP DES/TDES basic chaining modes (ECB, CBC).

*Note:* The outputs of the intermediate DEA stages are never revealed outside the cryptographic boundary, with the exclusion of the IV registers in CBC mode.

DES keying and chaining modes

The TDES allows three different keying options:

- **Three independent keys**
  
  The first option specifies that all the keys are independent, that is, K1, K2 and K3 are independent. FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to this option as the Keying Option 1 and, to the TDES as 3-key TDES.

- **Two independent keys**
  
  The second option specifies that K1 and K2 are independent and K3 is equal to K1, that is, K1 and K2 are independent, K3 = K1. FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to this option as the Keying Option 2 and, to the TDES as 2-key TDES.
– 1998) refers to this second option as the Keying Option 2 and, to the TDES as 2-key TDES.

- **Three equal keys**
  The third option specifies that K1, K2 and K3 are equal, that is:
  \[ K1 = K2 = K3 \]
  FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to the third option as the Keying Option 3. This “1-key” TDES is equivalent to single DES.

The following chaining algorithms are supported by the DES hardware and can be selected through the ALGOMODE bits in the CRYP_CR register:

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)

These modes are described in details in **Section 39.3.10: CRYP DES/TDES basic chaining modes (ECB, CBC).**

**39.3.4 CRYP AES cryptographic core**

**Overview**

The AES cryptographic core consists of the following components:

- The AES Algorithm (AEA core)
- The Multiplier over a binary Galois field (GF2mul)
- The key information
- The initialization vector (IV) or Nonce information
- Chaining algorithms logic (XOR, feedback/counter, mask)

The AES core works on 128-bit data blocks of (four words) with 128-, 192- or 256-bit key lengths. Depending on the chaining mode, the peripheral requires zero or one 128-bit initialization vector (IV).

The cryptographic peripheral features two operating modes:

- **ALGODIR=0**: Plaintext encryption using the key stored in the CRYP_Kx registers.
- **ALGODIR=1**: Ciphertext decryption using the key stored in the CRYP_Kx registers.
  When ECB and CBC chaining modes are selected, an initial key derivation process is automatically performed by the cryptographic peripheral.

The operating mode is selected by programming the ALGODIR bit in the CRYP_CR register.

**Typical data processing**

A description of cryptographic processor typical usage in AES mode can be found in **Section 39.3.11: CRYP AES basic chaining modes (ECB, CBC).**

**Note:** The outputs of the intermediate AEA stages is never revealed outside the cryptographic boundary, with the exclusion of the IV registers.
AES chaining modes

The following chaining algorithms are supported by the cryptographic processor and can be selected through the ALGOMODE bits in the CRYP_CR register:

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Counter Mode (CTR)
- Galois/Counter Mode (GCM)
- Galois Message Authentication Code mode (GMAC)
- Counter with CBC-MAC (CCM)

A quick introduction on these chaining modes can be found in the following subsections.

For detailed instructions, refer to Section 39.3.11: CRYP AES basic chaining modes (ECB, CBC) and onward.

AES Electronic CodeBook (ECB)

ECB is the simplest operating mode. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

Note: For decryption, a special key scheduling is required before processing the first block.
AES Cipher block chaining (CBC)

CBC operating mode chains the output of each block with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

Note: For decryption, a special key scheduling is required before processing the first block.
AES Counter mode (CTR)

The CTR mode uses the AES core to generate a key stream; these keys are then XORed with the plaintext to obtain the ciphertext as specified in NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.

**Note:** Unlike ECB and CBC modes, no key scheduling is required for the CTR decryption, since in this chaining scheme the AES core is always used in encryption mode for producing the counter blocks.
In Galois/Counter mode (GCM), the plaintext message is encrypted, while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. It requires an initial value and a particular 128-bit block at the end of the message.

AES Galois Message Authentication Code (GMAC)
Galois Message Authentication Code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.

GMAC is similar to Galois/Counter mode (GCM), except that it is applied on a message composed only by clear-text authenticated data (i.e. only header, no payload).

**AES Counter with CBC-MAC (CCM)**

In Counter with Cipher Block Chaining-Message Authentication Code (CCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and the corresponding MAC (also known as tag). It is described by NIST in Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality.

CCM mode is based on AES in counter mode for confidentiality and it uses CBC for computing the message authentication code. It requires an initial value.

Like GCM CCM chaining mode, AES-CCM mode can be applied on a message composed only by cleartext authenticated data (i.e. only header, no payload). Note that this way of using CCM is not called CMAC (it is not similar to GCM/GMAC), and its usage is not recommended by NIST.
39.3.5 CRYP procedure to perform a cipher operation

Introduction

To understand how the cryptographic peripheral operates, a typical cipher operation is described below. For the detailed peripheral usage according to the cipher mode, refer to the specific section, e.g. *Section 39.3.11: CRYP AES basic chaining modes (ECB, CBC)*.

CRYP initialization

1. Initialize the cryptographic processor. The order of operations is not important except for AES-ECB and AES-CBC decryption, where the key preparation requires a specific sequence.
   a) Disable the cryptographic processor by setting to 0 the CRYPEN bit in the CRYP_CR register.
   b) Configure the key size (128-, 192- or 256-bit, in the AES only) with the KEYSIZE bits in the CRYP_CR register.
   c) Write the symmetric key into the CRYP_KxL/R registers (2 to 8 registers to be written depending on the algorithm).
   d) Configure the data type (1-, 8-, 16- or 32-bit), with the DATATYPE bits in the CRYP_CR register.
   e) In case of decryption in AES-ECB or AES-CBC mode, prepare the key that has been written. First configure the key preparation mode by setting the ALGOMODE bits to 0b111 in the CRYP_CR register. Then write the CRYPEN bit to 1: the BUSY bit is set automatically. Wait until BUSY returns to 0 (CRYPEN is automatically cleared as well): the key is prepared for decryption.
   f) Configure the algorithm and chaining with the ALGOMODE bits in the CRYP_CR register.
   g) Configure the direction (encryption/decryption) through the ALGODIR bit in the CRYP_CR register.
   h) When it is required (e.g. CBC or CTR chaining modes), write the initialization vectors into the CRYP_IVxL/R register.
2. Flush the IN and OUT FIFOs by writing the FFLUSH bit to 1 in the CRYP_CR register.

Preliminary warning for all cases

If the ECB or CBC mode is selected and data are not a multiple of 64 bits (for DES) or 128 bits (for AES), the second and the last block management is more complex than the sequences below. Refer to *Section 39.3.8: CRYP stealing and data padding* for more details.
Appending data using the CPU in polling mode
1. Enable the cryptographic processor by setting to 1 the CRYPEN bit in the CRYP_CR register.
2. Write data in the IN FIFO (one block or until the FIFO is full).
3. Repeat the following sequence until the second last block of data has been processed:
   a) Wait until the not-empty-flag OFNE is set to 1, then read the OUT FIFO (one block or until the FIFO is empty).
   b) Wait until the not-full-flag IFNF is set to 1, then write the IN FIFO (one block or until the FIFO is full) except if it is the last block.
4. The BUSY bit is set automatically by the cryptographic processor. At the end of the processing, the BUSY bit returns to 0 and both FIFOs are empty (IN FIFO empty flag IFEM=1 and OUT FIFO not empty flag OFNE=0).
5. If the next processing block is the last block, the CPU must pad (when applicable) the data with zeroes to obtain a complete block
6. When the operation is complete, the cryptographic processor can be disabled by clearing the CRYPEN bit in CRYP_CR register.

Appending data using the CPU in interrupt mode
1. Enable the interrupts by setting the INIM and OUTIM bits in the CRYP_IMSCR register.
2. Enable the cryptographic processor by setting to 1 the CRYPEN bit in the CRYP_CR register.
3. In the interrupt service routine that manages the input data:
   a) If the last block is being loaded, the CPU must pad (when applicable) the data with zeroes to have a complete block. Then load the block into the IN FIFO.
   b) If it is not the last block, load the data into the IN FIFO. You can load only one block (2 words for DES, 4 words for AES), or load data until the FIFO is full.
   c) In all cases, after the last word of data has been written, disable the interrupt by clearing the INIM interrupt mask.
4. In the interrupt service routine that manages the input data:
   a) Read the output data from the OUT FIFO. You can read only one block (2 words for DES, 4 words for AES), or read data until the FIFO is empty.
   b) When the last word has been read, INIM and BUSY bits are set to 0 and both FIFOs are empty (IFEM=1 and OFNE=0). You can disable the interrupt by clearing the OUTIM bit, and disable the peripheral by clearing the CRYPEN bit.
   c) If you read the last block of cleartext data (i.e. decryption), optionally discard the data that is not part of message/payload.

Appending data using the DMA
1. Prepare the last block of data by optionally padding it with zeroes to have a complete block.
2. Configure the DMA controller to transfer the input data from the memory and transfer the output data from the peripheral to the memory, as described in Section 39.3.19:
**CRYP DMA interface.** The DMA should be configured to set an interrupt on transfer completion to indicate that the processing is complete.

3. Enable the cryptographic processor by setting to 1 the CRYPEN bit in CRYP_CR register, then enable the DMA IN and OUT requests by setting to 1 the DIEN and DOEN bits in the CRYP_DMACR register.

4. All the transfers and processing are managed by the DMA and the cryptographic processor. The DMA interrupt indicates that the processing is complete. Both FIFOs are normally empty and BUSY flag is set 0.

**Caution:** If the application needs to suspend a message to process another one with a higher priority, refer to Section 39.3.9: CRYP suspend/resume operations

When a block is being processed in DES or TDES mode, if the output FIFO becomes full and the input FIFO contains at least one new block, then the new block is popped off the input FIFO and the BUSY bit remains high until there is enough space to store this new block into the output FIFO.

**39.3.6 CRYP busy state**

The cryptographic processor is busy and processing data (BUSY set to 1 in CRYP_SR register) when all the conditions below are met:

- CRYPEN = 1 in CRYP_CR register.
- There are enough data in the input FIFO (at least two words for the DES or TDES algorithm mode, four words for the AES algorithm mode).
- There is enough free-space in the output FIFO (at least two word locations for DES, four for AES).

Write operations to the CRYP_Kx(L/R)R key registers, to the CRYP_IVx(L/R)R initialization registers, or to bits [9:2] of the CRYP_CR register, are ignored when cryptographic processor is busy (i.e. the registers are not modified). It is thus not possible to modify the configuration of the cryptographic processor while it is processing a data block.

It is possible to clear the CRYPEN bit while BUSY bit is set to 1. In this case the ongoing DES/TDES or AES processing first completes (i.e. the word results are written to the output FIFO) before the BUSY bit is cleared by hardware.

**Note:** If the application needs to suspend a message to process another one with a higher priority, refer to Section 39.3.9: CRYP suspend/resume operations

When a block is being processed in DES or TDES mode, if the output FIFO becomes full and the input FIFO contains at least one new block, then the new block is popped off the input FIFO and the BUSY bit remains high until there is enough space to store this new block into the output FIFO.

**39.3.7 Preparing the CRYP AES key for decryption**

When performing an AES ECB or CBC decryption, the AES key has to be prepared, i.e. a complete key schedule of encryption is required before performing the decryption. In other words, the key in the last round of encryption must be used as the first round key for decryption.

This preparation is not required in any other AES modes than AES ECB or CBC decryption.

If the application software stores somehow the initial key prepared for decryption, the key scheduling operation can be performed only once for all the data to be decrypted with a given cipher key.
**Note:** The latency of the key preparation operation is 14, 16 or 18 clock cycles depending on the key size (128-, 192- or 256-bit).

The CRYP key preparation process is performed as follows:

1. Write the encryption key to K0...K3 key registers.
2. Program ALGOMODE bits to 0b111 in CRYP_CR. Writing this value when CRYPEN is set to 1 immediately starts an AES round for key preparation. The BUSY bit in the CRYP_SR register is set to 1.
3. When the key processing is complete, the resulting key is copied back into the K0...K3 key registers, and the BUSY bit is cleared.

**Note:** As the CRYPEN bitfield is reset by hardware at the end of the key preparation, the application software must set it again for the next operation.

### 39.3.8 CRYP stealing and data padding

When using DES or AES algorithm in **ECB** or **CBC** modes to manage messages that are not multiple of the block size (64 bits for DES, 128 bits for AES), use ciphertext stealing techniques such as those described in NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode. Since the cryptographic processor does not implement such techniques, the last two blocks must be handled in a special way by the application.

**Note:** Ciphertext stealing techniques are not documented in this reference manual.

Similarly, when the AES algorithm is used in other modes than ECB or CBC, incomplete input data blocks (i.e. block shorter than 128 bits) have to be padded with zeroes by the application prior to encryption (i.e. extra bits should be appended to the trailing end of the data string). After decryption, the extra bits have to be discarded. The cryptographic processor does not implement automatic data padding operation to the last block, so the application should follow the recommendation given in Section 39.3.5: CRYP procedure to perform a cipher operation to manage messages that are not multiple of 128 bits.

**Note:** Padding data are swapped in a similar way as normal data, according to the DATATYPE field in CRYP_CR register (see Section 39.3.16: CRYP data registers and data swapping for details).
With this version of cryptographic processor, a special workaround is required in order to properly compute authentication tags while doing a **GCM encryption** or a **CCM decryption** with the last block of payload size **inferior to 128 bits**. This workaround is described below:

- **During GCM encryption payload phase** and before inserting a last plaintext block smaller than 128 bits, the application has to follow the below sequence:
  a) Disable the peripheral by setting the CRYPEN bit to 0 in CRYP_CR.
  b) Load CRYP_IV1R register content in a temporary variable. Decrement the value by 1 and reinsert the result in CRYP_IV1R register.
  c) Change the AES mode to CTR mode by writing the ALGOMODE bitfield to 0b0110 in the CRYP_CR register.
  a) Set the CRYPEN bit to 1 in CRYP_CR to enable again the peripheral.
  b) Pad the last block (smaller than 128 bits) with zeros to have a complete block of 128 bits, then write it into CRYP_DIN register.
  c) Upon encryption completion, read the 128-bit generated ciphertext from the CRYP_DOUT register and store it as intermediate data.
  d) Change again the AES mode to GCM mode by writing the ALGOMODE bitfield to 0b1000 in the CRYP_CR register.
  e) Select Final phase by writing the GCM_CCMPH bitfield to 0b11 in the CRYP_CR register.
  f) In the intermediate data, set to 0 the bits corresponding to the padded bits of the last payload block then insert the resulting data to CRYP_DIN register.
  g) When the operation is complete, read data from CRYP_DOUT. These data have to be discarded.
  h) Apply the normal Final phase as described in Section 39.3.13: **CRYP AES Galois/counter mode (GCM)**.

- **During CCM decryption payload phase** and before inserting a last ciphertext block smaller than 128 bits, the application has to follow the below sequence:
  a) To disable the peripheral, set the CRYPEN bit to 0 in CRYP_CR.
  b) Load CRYP_IV1R in a temporary variable (named here IV1temp).
  c) Load CRYP_CSGCMCCM0R, CRYP_CSGCMCCM1R, CRYP_CSGCMCCM2R, and CRYP_CSGCMCCM3R register contents from LSB to MSB in 128-bit temporary variable (named here temp1).
  d) Load in CRYP_IV1R the content previously stored in IV1temp.
  e) Change the AES mode to CTR mode by writing the ALGOMODE bitfield to 0b0110 in the CRYP_CR register.
  a) Set the CRYPEN bit to 1 in CRYP_CR to enable again the peripheral.
  b) Pad the last block (smaller than 128 bits) with zeros to have a complete block of 128 bits, then write it to CRYP_DIN register.
  c) Upon decryption completion, read the 128-bit generated data from DOUT register, and store them as intermediate data (here named intdata_o).
  d) Save again CRYP_CSGCMCCM0R, CRYP_CSGCMCCM1R, CRYP_CSGCMCCM2R, and CRYP_CSGCMCCM3R register contents, from LSB to MSB, in a new 128-bit temporary variable (named here temp2).
  e) Change again the AES mode to CCM mode by writing the ALGOMODE bitfield to 0b1001 in the CRYP_CR register.
f) Select the header phase by writing the GCM_CCMPH bitfield to 0b01 in the CRYP_CR register.

g) In the intermediate data (intdata_o which was generated with CTR), set to 0 the bits corresponding to the padded bits of the last payload block, XOR with temp1, XOR with temp2, and insert the resulting data into CRYP_DIN register. In other words:

\[
\text{CRYP_DIN} = (\text{intdata}_o \text{ AND mask}) \text{ XOR temp1 XOR temp2.}
\]

h) Wait for operation completion.

i) Apply the normal Final phase as described in Section 39.3.15: CRYP AES Counter with CBC-MAC (CCM).

### 39.3.9 CRYP suspend/resume operations

A message can be suspended if another message with a higher priority has to be processed. When this highest priority message has been sent, the suspended message can be resumed in both encryption or decryption mode.

Suspend/resume operations do not break the chaining operation and the message processing can be resumed as soon as cryptographic processor is enabled again to receive the next data block.

*Figure 288* gives an example of suspend/resume operation: message 1 is suspended in order to send a higher priority message (message 2), which is shorter than message 1 (AES algorithm).

*Figure 288. Example of suspend mode management*

A detailed description of suspend/resume operations can be found in each AES mode section.
39.3.10 **CRYP DES/TDES basic chaining modes (ECB, CBC)**

**Overview**

FIPS PUB 46-3 – 1999 (and ANSI X9.52-1998) provides a thorough explanation of the processing involved in the four operation modes supplied by the DES computing core: TDES-ECB encryption, TDES-ECB decryption, TDES-CBC encryption and TDES-CBC decryption. This section only gives a brief explanation of each mode.

**DES/TDES-ECB encryption**

*Figure 289* illustrates the encryption in DES and TDES Electronic CodeBook (DES/TDES-ECB) mode. This mode is selected by writing in ALGOMODE to 0b000 and ALGODIR to 0 in CRYP_CR.

![Figure 289. DES/TDES-ECB mode encryption](image)

1. K: key; C: cipher text; I: input block; O: output block; P: plain text.

A 64-bit plaintext data block (P) is used after bit/byte/half-word as the input block (I). The input block is processed through the DEA in the encrypt state using K1. The output of this process is fed back directly to the input of the DEA where the DES is performed in the decrypt state using K2. The output of this process is fed back directly to the input of the DEA where the DES is performed in the encrypt state using K3. The resultant 64-bit output block (O) is used, after bit/byte/half-word swapping, as ciphertext (C) and it is pushed into the OUT FIFO.

*Note:* For more information on data swapping, refer to Section 39.3.16: CRYP data registers and data swapping.

Detailed DES/TDES encryption sequence can be found in Section 39.3.5: CRYP procedure to perform a cipher operation.
DES/TDES-ECB mode decryption

*Figure 290* illustrates the decryption in DES and TDES Electronic CodeBook (DES/TDES-ECB) mode. This mode is selected by writing ALGOMODE to 0b000 and ALGODIR to 1 in CRYP_CR.

**Figure 290. DES/TDES-ECB mode decryption**

A 64-bit ciphertext block (C) is used, after bit/byte/half-word swapping, as the input block (I). The keying sequence is reversed compared to that used in the encryption process. The input block is processed through the DEA in the decrypt state using K3. The output of this process is fed back directly to the input of the DEA where the DES is performed in the encrypt state using K2. The new result is directly fed to the input of the DEA where the DES is performed in the decrypt state using K1. The resultant 64-bit output block (O), after bit/byte/half-word swapping, produces the plaintext (P).

**Note:** For more information on data swapping refer to Section 39.3.16: CRYP data registers and data swapping.

Detailed DES/TDES encryption sequence can be found in Section 39.3.5: CRYP procedure to perform a cipher operation.
DES/TDES-CBC encryption

Figure 291 illustrates the encryption in DES and TDES Cipher Block Chaining (DES/TDES-ECB) mode. This mode is selected by writing in ALGOMODE to 0b001 and ALGODIR to 0 in CRYP_CR.

Figure 291. DES/TDES-CBC mode encryption

K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: initialization vectors.

This mode begins by dividing a plaintext message into 64-bit data blocks. In TCBC encryption, the first input block (I1), obtained after bit/byte/half-word swapping, is formed by exclusive-ORing the first plaintext data block (P1) with a 64-bit initialization vector IV (I1 = IV ⊕ P1). The input block is processed through the DEA in the encrypt state using K1. The output of this process is fed back directly to the input of the DEA, which performs the DES in the decrypt state using K2. The output of this process is fed directly to the input of the DEA, which performs the DES in the encrypt state using K3. The resultant 64-bit output block (O1) is used directly as the ciphertext (C1), that is, C1 = O1.

This first ciphertext block is then exclusive-ORed with the second plaintext data block to produce the second input block, (I2) = (C1 ⊕ P2). Note that I2 and P2 now refer to the second block. The second input block is processed through the TDEA to produce the second ciphertext block.

This encryption process continues to “chain” successive cipher and plaintext blocks together until the last plaintext block in the message is encrypted.

If the message does not consist of an integral number of data blocks, then the final partial data block should be encrypted in a manner specified for the application.
Note: For more information on data swapping refer to Section 39.3.16: CRYP data registers and data swapping.

Detailed DES/TDES encryption sequence can be found in Section 39.3.5: CRYP procedure to perform a cipher operation.

**DES/TDES-CBC decryption**

*Figure 291* illustrates the decryption in DES and TDES Cipher Block Chaining (DES/TDES-ECB) mode. This mode is selected by writing ALGOMODE to 0b001 and ALGODIR to 1 in CRYP_CR.

**Figure 292. DES/TDES-CBC mode decryption**

In this mode the first ciphertext block (C₁) is used directly as the input block (I₁). The keying sequence is reversed compared to that used for the encrypt process. The input block is processed through the DEA in the decrypt state using K₃. The output of this process is fed directly to the input of the DEA where the DES is processed in the encrypt state using K₂. This resulting value is directly fed to the input of the DEA where the DES is processed in the decrypt state using K₁. The resulting output block is exclusive-ORed with the IV (which must be the same as that used during encryption) to produce the first plaintext block (P₁ = O₁ ⊕ IV).

The second ciphertext block is then used as the next input block and is processed through the TDEA. The resulting output block is exclusive-ORed with the first ciphertext block to produce the second plaintext data block (P₂ = O₂ ⊕ C₁). Note that P₂ and O₂ refer to the second block of data.
The DES/TDES-CBC decryption process continues in this manner until the last complete ciphertext block has been decrypted.

Ciphertext representing a partial data block must be decrypted in a manner specified for the application.

Note: For more information on data swapping refer to Section 39.3.16: CRYP data registers and data swapping.

Detailed DES/TDES encryption sequence can be found in Section 39.3.5: CRYP procedure to perform a cipher operation.

DES/TDES suspend/resume operations in ECB/CBC modes

Before interrupting the current message, the user application must respect the following steps:

1. If DMA is used, stop the DMA transfers to the IN FIFO by clearing to 0 the DIEN bit in the CRYP_DMACR register.
2. Wait until both the IN and the OUT FIFOs are empty (IFEM=1 and OFNE=0 in the CRYP_SR) and the BUSY bit is cleared. Alternatively, as the input FIFO can contain up to four unprocessed DES blocks, the application could decide for real-time reason to interrupt the cryptographic processing without waiting for the IN FIFO to be empty. In this case, the alternative is:
   a) Wait until OUT FIFO is empty (OFNE=0).
   b) Read back the data loaded in the IN FIFO that have not been processed and save them in the memory until the IN FIFO is empty.
3. If DMA is used stop the DMA transfers from the OUT FIFO by clearing to 0 the DOEN bit in the CRYP_DMACR register.
4. Disable the cryptographic processor by setting the CRYPEN bit to 0 in CRYP_CR, then save the current configuration (bits [9:2] in the CRYP_CR register). If CBC mode is selected, save the initialization vector registers, since CRYP_IVx registers have changed from initial values during the data processing.

Note: Key registers do not need to be saved as the original key value is known by the application.

5. If DMA is used, save the DMA controller status (such as the pointers to IN and OUT data transfers, number of remaining bytes).

To resume message processing, the user application must respect the following sequence:

1. If DMA is used, reconfigure the DMA controller to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Make sure the cryptographic processor is disabled by reading the CRYPEN bit in CRYP_CR (it must be 0).
3. Configure again the cryptographic processor with the initial setting in CRYP_CR, as well as the key registers using the saved configuration.
4. If the CBC mode is selected, restore CRYP_IVx registers using the saved configuration.
5. Optionally, write the data that were saved during context saving into the IN FIFO.
6. Enable the cryptographic processor by setting the CRYPEN bit to 1.
7. If DMA is used, enable again DMA requests for the cryptographic processor, by setting to 1 the DIEN and DOEN bits in the CRYP_DMACR register.
39.3.11 CRYP AES basic chaining modes (ECB, CBC)

Overview

FIPS PUB 197 (November 26, 2001) provides a thorough explanation of the processing involved in the four basic operation modes supplied by the AES computing core: AES-ECB encryption, AES-ECB decryption, AES-CBC encryption and AES-CBC decryption. This section only gives a brief explanation of each mode.

AES ECB encryption

Figure 293 illustrates the AES Electronic codebook (AES-ECB) mode encryption. This mode is selected by writing ALGOMODE to 0b100 and ALGODIR to 0 in CRYP_CR.

![Figure 293. AES-ECB mode encryption](image)

1. K: key; C: cipher text; I: input block; O: output block; P: plain text.
2. If Key size = 128: Key = \([K_3 \ K_2]\).
   - If Key size = 192: Key = \([K_3 \ K_2 \ K_1]\).
   - If Key size = 256: Key = \([K_3 \ K_2 \ K_1 \ K_0]\).

In this mode a 128-bit plaintext data block (P) is used after bit/byte/half-word swapping as the input block (I). The input block is processed through the AEA in the encrypt state using the 128, 192 or 256-bit key. The resultant 128-bit output block (O) is used after bit/byte/half-word swapping as ciphertext (C). It is then pushed into the OUT FIFO.

For more information on data swapping refer to Section 39.3.16: CRYP data registers and data swapping.
AES ECB decryption

*Figure 294* illustrates the AES Electronic codebook (AES-ECB) mode decryption. This mode is selected by writing in ALGOMODE to 0b100 and ALGODIR to 1 in CRYP_CR.

**Figure 294. AES-ECB mode decryption**

1. K: key; C: cipher text; I: input block; O: output block; P: plain text.
2. If Key size = 128 => Key = [K3 K2].
   If Key size = 192 => Key = [K3 K2 K1].
   If Key size = 256 => Key = [K3 K2 K1 K0].

To perform an AES decryption in ECB mode, the secret key has to be prepared (it is necessary to execute the complete key schedule for encryption) by collecting the last round key, and using it as the first round key for the decryption of the ciphertext. This preparation phase is computed by the AES core. Refer to *Section 39.3.7: Preparing the CRYP AES key for decryption* for more details on how to prepare the key.

When the key preparation is complete, the decryption proceed as follow: a 128-bit ciphertext block (C) is used after bit/byte/half-word swapping as the input block (I). The keying sequence is reversed compared to that of the encryption process. The resultant 128-bit output block (O), after bit/byte or half-word swapping, produces the plaintext (P). The AES-CBC decryption process continues in this manner until the last complete ciphertext block has been decrypted.

For more information on data swapping refer to *Section 39.3.16: CRYP data registers and data swapping.*
AES CBC encryption

Figure 295 illustrates the AES Cipher block chaining (AES-CBC) mode encryption. This mode is selected by writing ALGOMODE to 0b101 and ALGODIR to 0 in CRYP_CR.

In this mode the first input block (I₁) obtained after bit/byte/half-word swapping is formed by exclusive-ORing the first plaintext data block (P₁) with a 128-bit initialization vector IV (I₁ = IV ⊕ P₁). The input block is processed through the AEA in the encrypt state using the 128-, 192- or 256-bit key (K₀...K₃). The resultant 128-bit output block (O₁) is used directly as ciphertext (C₁), that is, C₁ = O₁. This first ciphertext block is then exclusive-ORed with the second plaintext data block to produce the second input block, (I₂) = (C₁ ⊕ P₂). Note that I₂ and P₂ now refer to the second block. The second input block is processed through the AEA to produce the second ciphertext block. This encryption process continues to “chain” successive cipher and plaintext blocks together until the last plaintext block in the message is encrypted.

If the message does not consist of an integral number of data blocks, then the final partial data block should be encrypted in a manner specified for the application, as explained in Section 39.3.8: CRYP stealing and data padding.

For more information on data swapping, refer to Section 39.3.16: CRYP data registers and data swapping.
**AES CBC decryption**

*Figure 296* illustrates the AES Cipher block chaining (AES-CBC) mode decryption. This mode is selected by writing ALGOMODE to 0b101 and ALGODIR to 1 in CRYP.CR.

**Figure 296. AES-CBC mode decryption**

1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: Initialization vectors.

2. IVx=[IVxR IVxL], R=right, L=left.

3. If Key size = 128 => Key = [K3 K2].
   - If Key size = 192 => Key = [K3 K2 K1].
   - If Key size = 256 => Key = [K3 K2 K1 K0].

In CBC mode, like in ECB mode, the secret key must be prepared to perform an AES decryption. Refer to Section 39.3.7: Preparing the CRYP AES key for decryption for more details on how to prepare the key.

When the key preparation process is complete, the decryption proceeds as follow: the first 128-bit ciphertext block (C1) is used directly as the input block (I1). The input block is processed through the AEA in the decrypt state using the 128-, 192- or 256-bit key. The resulting output block is exclusive-ORed with the 128-bit initialization vector IV (which must be the same as that used during encryption) to produce the first plaintext block (P1 = O1 ⊕ IV).

The second ciphertext block is then used as the next input block and is processed through the AEA. The resulting output block is exclusive-ORed with the first ciphertext block to produce the second plaintext data block (P2 = O2 ⊕ C1). Note that P2 and O2 refer to the second block of data. The AES-CBC decryption process continues in this manner until the last complete ciphertext block has been decrypted.
Ciphertext representing a partial data block must be decrypted in a manner specified for the application, as explained in Section 39.3.8: CRYP stealing and data padding.

For more information on data swapping, refer to Section 39.3.16: CRYP data registers and data swapping.

AES suspend/resume operations in ECB/CBC modes

Before interrupting the current message, the user application must respect the following sequence:

1. If DMA is used, stop the DMA transfers to the IN FIFO by clearing to 0 the DIEN bit in the CRYP_DMACR register.
2. Wait until both the IN and the OUT FIFOs are empty (IFEM=1 and OFNE=0 in the CRYP_SR) and the BUSY bit is cleared.
3. If DMA is used, stop the DMA transfers from the OUT FIFO by clearing to 0 the DOEN bit in the CRYP_DMACR register.
4. Disable the CRYP by setting the CRYPEN bit to 0 in CRYP_CR, then save the current configuration (bits [9:2] in the CRYP_CR register). If ECB mode is not selected, save the initialization vector registers, because CRYP_IVx registers have changed from initial values during the data processing.

**Note:** Key registers do not need to be saved as the original key value is known by the application.

5. If DMA is used, save the DMA controller status (such as pointers to IN and OUT data transfers, number of remaining bytes).

To resume message processing, the user application must respect the following sequence:

1. If DMA is used, reconfigure the DMA controller to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Make sure the cryptographic processor is disabled by reading the CRYPEN bit in CRYP_CR (it must be set to 0).
3. Configure the cryptographic processor again with the initial setting in CRYP_CR, as well as the key registers using the saved configuration.
4. For AES-ECB or AES-CBC decryption, the key must be prepared again, as described in Section 39.3.7: Preparing the CRYP AES key for decryption.
5. If ECB mode is not selected, restore CRYP_IVx registers using the saved configuration.
6. Enable the cryptographic processor by setting the CRYPEN bit to 1.
7. If DMA is used, enable again the DMA requests from the cryptographic processor, by setting DIEN and DOEN bits to 1 in the CRYP_DMACR register.
39.3.12 CRYP AES counter mode (AES-CTR)

Overview

The AES counter mode (CTR) uses the AES block as a key stream generator. The generated keys are then XORed with the plaintext to obtain the ciphertext.

CTR chaining is defined in NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation. A typical message construction in CTR mode is given in Figure 297.

Figure 297. Message construction for the Counter mode

The structure of this message is as below:

- A 16-byte Initial Counter Block (ICB), composed of three distinct fields:
  - A nonce: a 32-bit, single-use value (i.e. a new nonce should be assigned to each new communication).
  - The initialization vector (IV): a 64-bit value that must be unique for each execution of the mode under a given key.
  - The counter: a 32-bit big-endian integer that is incremented each time a block has been processed. The initial value of the counter should be set to 1.
- The plaintext (P) is both authenticated and encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.
AES CTR processing

*Figure 298* (respectively *Figure 299*) describes the AES-CTR encryption (respectively decryption) process implemented within this peripheral. This mode is selected by writing in ALGOMODE bitfield to 0b110 in CRYP_CR.

**Figure 298. AES-CTR mode encryption**

1. K: key; C: cipher text; I: input Block; o: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); Cs: cipher text after swapping (when decoding) or before swapping (when encoding); P: plain text; IV: Initialization vectors.
1. K: key; C: cipher text; I: input Block; o: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); Cs: cipher text after swapping (when decoding) or before swapping (when encoding); P: plain text; IV: Initialization vectors.

In CTR mode, the output block is XORed with the subsequent input block before it is input to the algorithm. Initialization vectors in the peripheral must be initialized as shown on Table 280.

**Table 280. Counter mode initialization vector**

<table>
<thead>
<tr>
<th>CRYP_IV1R[31:0]</th>
<th>CRYP_IV1L[31:0]</th>
<th>CRYP_IV0R[31:0]</th>
<th>CRYP_IV0L[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonce</td>
<td>IV[63:32]</td>
<td>IV[31:0]</td>
<td>32-bit counter= 0x1</td>
</tr>
</tbody>
</table>

Unlike in CBC mode, which uses the CRYP_IVx registers only once when processing the first data block, in CTR mode IV registers are used for processing each data block, and the peripheral increments the least significant 32 bits (leaving the other most significant 96 bits unchanged).

CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that will be XORed with the plaintext or cipher as input. Thus when ALGOMODE is set to 0b110, ALGODIR is don’t care.

**Note:** In this mode the key must NOT be prepared for decryption.
The following sequence must be used to perform an encryption or a decryption in CTR chaining mode:

1. Make sure the cryptographic processor is disabled by clearing the CRYPEN bit in the CRYP_Cr register.
2. Configure CRYP_CR as follows:
   a) Program ALGOMODE bits to 0b110 to select CTR mode.
   b) Configure the data type (1, 8, 16 or 32 bits) through the DATATYPE bits.
3. Initialize the key registers (128, 192 and 256 bits) in CRYP_KEYRx as well as the initialization vector (IV) as described in Table 280.
4. Flush the IN and OUT FIFOs by writing the FFLUSH bit to 1 in the CRYP_Cr register.
5. If it is the last block, optionally pad the data with zeros to have a complete block.
6. Append data in the cryptographic processor and read the result. The three possible scenarios are described in Section 39.3.5: CRYP procedure to perform a cipher operation.
7. Repeat the previous step until the second last block is processed. For the last block, execute the two previous steps. For this last block, the driver must discard the data that is not part of the data when the last block size is less than 16 bytes.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher priority message, and resume the message which was interrupted. Detailed CBC sequence can be found in Section 39.3.11: CRYP AES basic chaining modes (ECB, CBC).

Note: Like for CBC mode, IV registers must be reloaded during the resume operation.
39.3.13 CRYP AES Galois/counter mode (GCM)

Overview

The AES Galois/counter mode (GCM) allows encrypting and authenticating the plaintext, and generating the correspondent ciphertext and tag (also known as message authentication code). To ensure confidentiality, GCM algorithm is based on AES counter mode. It uses a multiplier over a fixed finite field to generate the tag.

GCM chaining is defined in NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC. A typical message construction in GCM mode is given in Figure 300.

Figure 300. Message construction for the Galois/counter mode

The structure of this message is defined as below:

- A 16-byte Initial Counter Block (ICB), composed of two distinct fields:
  - The initialization vector (IV): a 96-bit value that must be unique for each execution of the mode under a given key. Note that the GCM standard supports IV that are shorter than 96-bit, but in this case strict rules apply.
  - The counter: a 32-bit big-endian integer that is incremented each time a block has been processed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.

- The authenticated header A (also knows as Additional Authentication Data) has a known length \( \text{Len}(A) \) that can be non-multiple of 16 bytes and cannot exceed \( 2^{64} - 1 \) bits. This part of the message is only authenticated, not encrypted.

- The plaintext message (P) is both authenticated and encrypted as ciphertext C, with a known length \( \text{Len}(P) \) that can be non-multiple of 16 bytes, and cannot exceed \( 2^{32} - 2 \) blocks of 128-bits.

Note: GCM standard specifies that ciphertext C has same bit length as the plaintext P.

- When a part of the message (AAD or P) has a length which is non-multiple of 16 bytes, a special padding scheme is required.
- The last block is composed of the length of A (on 64 bits) and the length of ciphertext C (on 64 bits) as shown in Table 281.
AES GCM processing

This mode is selected by writing ALGOMODE bitfield to 0b110 in CRYP_CR.

The mechanism for the confidentiality of the plaintext in GCM mode is a variation of the Counter mode, with a particular 32-bit incrementing function that generates the necessary sequence of counter blocks.

CRYP_IV registers are used for processing each data block. The cryptographic processor automatically increments the 32 least significant bits of the counter block. The first counter block (CB1) written by the application is equal to the Initial Counter Block incremented by one (see Table 282).

Note: In this mode the key must NOT be prepared for decryption.

The authentication mechanism in GCM mode is based on a hash function, called GF2mul, that performs multiplication by a fixed parameter, called the hash subkey (H), within a binary Galois field.

To process a GCM message, the driver must go through four phases, which are described in the following subsections.

- The Init phase: the peripheral prepares the GCM hash subkey (H) and performs the IV processing.
- The Header phase: the peripheral processes the Additional Authenticated Data (AAD), with hash computation only.
- The Payload phase: the peripheral processes the plaintext (P) with hash computation, keystream encryption and data XORing. It operates in a similar way for ciphertext (C).
- The Final phase: the peripheral generates the authenticated tag (T) using the data last block.

Table 281. GCM last block definition

<table>
<thead>
<tr>
<th>Endianness</th>
<th>Bit[0]</th>
<th>Bit[32]</th>
<th>Bit[64]</th>
<th>Bit[96]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input data</td>
<td>0x0</td>
<td>Header length[31:0]</td>
<td>0x0</td>
<td>Payload length[31:0]</td>
</tr>
</tbody>
</table>

Table 282. GCM mode IV registers initialization

<table>
<thead>
<tr>
<th>Register</th>
<th>CRYP_IV1R[31:0]</th>
<th>CRYP_IV1L[31:0]</th>
<th>CRYP_IV0R[31:0]</th>
<th>CRYP_IV0L[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32-bit counter= 0x2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. **GCM init phase**
   During this first step, the GCM hash subkey (H) is calculated and saved internally to be used for processing all the blocks. It is recommended to follow the sequence below:
   
a) Make sure the cryptographic processor is disabled by clearing the CRYPEN bit in the CRYP_CR register.
   
b) Select the GCM chaining mode by programming ALGOMODE bits to 0b01000 in CRYP_CR.
   
c) Configure GCM_CCMPH bits to 0b00 in CRYP_CR to indicate that the init phase is ongoing.
   
d) Initialize the key registers (128, 192 and 256 bits) in CRYP_KEYRx as well as the initialization vector (IV) as defined in Table 282.
   
e) Set CRYPEN bit to 1 to start the calculation of the hash key.
   
f) Wait for the CRYPEN bit to be cleared to 0 by the cryptographic processor, before moving on to the next phase.

2. **GCM header phase**
   The below sequence shall be performed after the GCM init phase. It must be complete before jumping to the payload phase. The sequence is identical for encryption and decryption.
   
g) Set the GCM_CCMPH bits to 0b01 in CRYP_CR to indicate that the header phase is ongoing.
   
h) Set the CRYPEN bit to 1 to start accepting data.
   
i) If it is the last block of additional authenticated data, optionally pad the data with zeros to have a complete block.
   
j) Append additional authenticated data in the cryptographic processor. The three possible scenarios are described in Section 39.3.5: CRYP procedure to perform a cipher operation.
   
k) Repeat the previous step until the second last additional authenticated data block is processed. For the last block, execute the two previous steps. Once all the additional authenticated data have been supplied, wait until the BUSY flag is cleared before moving on to the next phase.

**Note:** This phase can be skipped if there is no additional authenticated data, i.e. Len(A)=0.

In header and payload phases, CRYPEN bit is not automatically cleared by the cryptographic processor.
3. **GCM payload phase (encryption or decryption)**

When the payload size is not null, this sequence must be executed after the GCM header phase. During this phase, the encrypted/decrypted payload is stored in the CRYP_DOUT register.

- i) Set the CRYPEN bit to 0.
- m) Configure GCM_CCMPH to 0b10 in the CRYP_CR register to indicate that the payload phase is ongoing.
- n) Select the algorithm direction (0 for encryption, 1 for decryption) through the ALGODIR bit in CRYP_CR.
- o) Set the CRYPEN bit to 1 to start accepting data.
- p) If it is the last block of cleartext or plaintext, optionally pad the data with zeros to have a complete block. For encryption, refer to Section 39.3.8: CRYP stealing and data padding for more details.
- q) Append payload data in the cryptographic processor, and read the result. The three possible scenarios are described in Section 39.3.5: CRYP procedure to perform a cipher operation.
- r) Repeat the previous step until the second last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), execute the two previous steps. For the last block, the driver must discard the bits that are not part of the cleartext or the ciphertext when the last block size is less than 16 bytes. Once all payload data have been supplied, wait until the BUSY flag is cleared.

*Note:* This phase can be skipped if there is no payload data, i.e. Len(C)=0 (see GMAC mode).

4. **GCM final phase**

In this last step, the cryptographic processor generates the GCM authentication tag and stores it in CRYP_DOUT register.

- s) Configure GCM_CCMPH[1:0] to 0b11 in CRYP_CR to indicate that the Final phase is ongoing. Set the ALGODIR bit to 0 in the same register.
- t) Write the input to the CRYP_DIN register four times. The input must be composed of the length in bits of the additional authenticated data (coded on 64 bits) concatenated with the length in bits of the payload (coded of 64 bits), as show in Table 281.

*Note:* In this final phase data have to be swapped according to the DATATYPE programmed in CRYP_CR register.

- u) Wait until the OFNE flag (FIFO output not empty) is set to 1 in the CRYP_SR register.
- v) Read the CRYP_DOUT register four times: the output corresponds to the authentication tag.
- w) Disable the cryptographic processor (CRYPEN bit = 0 in CRYP_CR)
- x) If an authenticated decryption is being performed, compare the generated tag with the expected tag passed with the message.
Suspend/resume operations in GCM mode

Before interrupting the current message in header or payload phase, the user application must respect the following sequence:

1. If DMA is used, stop DMA transfers to the IN FIFO by clearing to 0 the DIEN bit in the CRYP_DMACR register.
2. Wait until both the IN and the OUT FIFOs are empty (IFEM=1 and OFNE=0 in the CRYP_SR register) and the BUSY bit is cleared.
3. If DMA is used, stop DMA transfers from the OUT FIFO by clearing to 0 the DOEN bit in the CRYP_DMACR register.
4. Disable the cryptographic processor by setting the CRYPEN bit to 0 in CRYP_CR, then save the current configuration (bits [9:2], bits [17:16] and bits 19 of the CRYP_CR register). In addition, save the initialization vector registers, since CRYP_IVx registers have changed from their initial values during data processing.

Note: Key registers do not need to be saved as original their key value is known by the application.

5. Save context swap registers: CRYP_CSGCMCCM0..7R and CRYP_CSGCM0..7R
6. If DMA is used, save the DMA controller status (pointers to IN and OUT data transfers, number of remaining bytes, etc.).

To resume message processing, the user must respect the following sequence:

1. If DMA is used, reconfigure the DMA controller to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Make sure the cryptographic processor is disabled by reading the CRYPEN bit in CRYP_CR (it must be 0).
3. Configure again the cryptographic processor with the initial setting in CRYP_CR, as well as the key registers using the saved configuration.
4. Restore context swap registers: CRYP_CSGCMCCM0..7R and CRYP_CSGCM0..7R
5. Restore CRYP_IVx registers using the saved configuration.
6. Enable the cryptographic processor by setting the CRYPEN bit to 1.
7. If DMA is used, enable again cryptographic processor DMA requests by setting to 1 the DIEN and DOEN bits in the CRYP_DMACR register.

Note: In Header phase, DMA OUT FIFO transfer is not used.
39.3.14 CRYP AES Galois message authentication code (GMAC)

Overview

The Galois message authentication code (GMAC) allows authenticating a plaintext and generating the corresponding tag information (also known as message authentication code). It is based on GCM algorithm, as defined in NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.

A typical message construction in GMAC mode is given in Figure 301.

Figure 301. Message construction for the Galois Message Authentication Code mode

AES GMAC processing

This mode is selected by writing ALGOMODE bitfield to 0b110 in CRYP.CR.

GMAC algorithm corresponds to the GCM algorithm applied on a message composed only of an header. As a consequence, all steps and settings are the same as in GCM mode, except that the payload phase (3) is not used.

Suspend/resume operations in GMAC

GMAC is exactly the same as GCM algorithm except that only header phase (2) can be interrupted.
39.3.15 CRYP AES Counter with CBC-MAC (CCM)

Overview

The AES Counter with Cipher Block Chaining-Message Authentication Code (CCM) algorithm allows encrypting and authenticating the plaintext, and generating the correspondent ciphertext and tag (also known as message authentication code). To ensure confidentiality, CCM algorithm is based on AES counter mode. It uses Cipher Block Chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve this CBC-MAC as an authentication mode outside of the context of the CCM specification.

CCM chaining is specified in NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality. A typical message construction in CCM mode is given in Figure 302.

Figure 302. Message construction for the Counter with CBC-MAC mode

![Figure 302. Message construction for the Counter with CBC-MAC mode](image)

The structure of this message is as below:

- One 16-byte first authentication block (called B0 by the standard), composed of three distinct fields:
  - **Q**: a bit string representation of the byte length of P (Plen)
  - A **nonce** (N): single-use value (i.e. a new nonce should be assigned to each new communication). Size of nonce Nlen + size of Plen shall be equal to 15 bytes.
  - **Flags**: most significant byte containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values t (MAC length expressed in bytes) and q (plaintext length such as Plen<2^{8q} bytes). Note that the counter blocks range associated to q is equal to 2^{8q-4}, i.e. if q maximum value is 8, the counter blocks used in cipher shall be on 60 bits.

Note: The cryptographic peripheral can only manage padded plaintext/ciphertext messages of length Plen < 2^{36} +1 bytes.
• 16-bytes blocks (B) associated to the Associated Data (A).
This part of the message is only authenticated, not encrypted. This section has a known length, \( ALen \), that can be a non-multiple of 16 bytes (see Figure 302). The standard also states that, on the MSB bits of the first message block (B1), the associated data length expressed in bytes (\( a \)) must be encoded as defined below:
  – If \( 0 < a < 2^{16} \cdot 2^8 \), then it is encoded as \([a]_{16}\), i.e. two bytes.
  – If \( 2^{16} \cdot 2^8 < a < 2^{32} \), then it is encoded as \(0xff \parallel 0xfe \parallel [a]_{32}\), i.e. six bytes.
  – If \( 2^{32} < a < 2^{64} \), then it is encoded as \(0xff \parallel 0xff \parallel [a]_{64}\), i.e. ten bytes.

• 16-byte blocks (B) associated to the plaintext message (P), which is both authenticated and encrypted as ciphertext C, with a known length of \( Plen \). This length can be a non-multiple of 16 bytes (see Figure 302) but cannot exceed \( 2^{32} \) blocks of 128-bit.

• The encrypted MAC (T) of length \( Tlen \) appended to the ciphertext C of overall length \( Clen \).

• When a part of the message (A or P) has a length which is a non-multiple of 16 bytes, a special padding scheme is required.

Note: \textit{CCM chaining mode can also be used with associated data only (i.e. no payload).}

As an example, the C.1 section in \textit{NIST Special Publication 800-38C} gives the following:

\( N: \) 10111213 141516 \quad (Nlen= 56 bits or 0x7 bytes)
\( A: \) 00010203 04050607 \quad (Alen= 64 bits or 0x8 bytes)
\( P: \) 20212223 \quad (Plen= 32 bits i.e. Q= 0x4 bytes)
\( T: \) 6084341b \quad (Tlen= 32 bits or t= 4)
\( B0: \) 4f101112 13141516 00000000 00000004
\( B1: \) 00080001 02030405 06070000 00000000
\( B2: \) 20212223 00000000 00000000 00000000
\( CTR0: \) 0710111213 141516 00000000 00000000
\( CTR1: \) 0710111213 141516 00000000 00000001

The usage of control blocks CTRx is explained in the following section. The generation of CTR0 from the first block (B0) must be managed by software.
AES CCM processing

This mode is selected by writing ALGOMODE bitfield to 0b1001 in CRYP_CR.

The data input to the generation-encryption process are a valid nonce, a valid payload string, and a valid associated data string, all properly formatted. The CBC chaining mechanism is applied to the formatted data to generate a MAC, whose length is known. Counter mode encryption, which requires a sufficiently long sequence of counter blocks as input, is applied to the payload string and separately to the MAC. The resulting data, called the ciphertext C, is the output of the generation-encryption process on plaintext P.

CRYP_IV registers are used for processing each data block. The cryptographic processor automatically increments the CTR counter with a bit length defined by the first block (B0). The first counter written by application, CTR1, is equal to B0 with the first 5 bits zeroed and the most significant bits containing P byte length also zeroed, then incremented by one (see Table 283).

Table 283. CCM mode IV registers initialization

<table>
<thead>
<tr>
<th>Register</th>
<th>CRYP_IV0L[31:0]</th>
<th>CRYP_IV0R[31:0]</th>
<th>CRYP_IV1L[31:0]</th>
<th>CRYP_IV1R[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input data</td>
<td>B0[31:0], where the 5 most significant bits are set to 0 (flag bits)</td>
<td>B0[63:32]</td>
<td>B0[95:64]</td>
<td>B0[127:96], where Q length bits are set to 0, except for bit 0 that is set to 1</td>
</tr>
</tbody>
</table>

Note: In this mode, the key must NOT be prepared for decryption.

To process a CCM message, the driver must go through four phases, which are described below:

- The Init phase: the peripheral processes the first block and prepares the first counter block.
- The Header phase: the peripheral processes the Associated data (A), with hash computation only.
- The Payload phase: the peripheral processes the plaintext (P), with hash computation, counter block encryption and data XORing. It operates in a similar way for ciphertext (C).
- The Final phase: the peripheral generates the message authentication code (MAC).

In this mode, the key must NOT be prepared for decryption.
1. **CCM init phase**

   In this first step, the first block (B0) of the CCM message is programmed into the CRYP_DIN register. During this phase, the CRYP_DOUT register does not contain any output data. It is recommended to follow the sequence below:
   
   a) Make sure that the cryptographic processor is disabled by clearing the CRYPEN bit in the CRYP_CR register.
   
   b) Select the CCM chaining mode by programming the ALGOMODE bits to 0b01001 in the CRYP_CR register.
   
   c) Configure the GCM_CCMPH bits to 0b00 in CRYP_CR to indicate that we are in the init phase.
   
   d) Initialize the key registers (128, 192 and 256 bits) in CRYP_KEYRx as well as the initialization vector (IV) with CTR1 information, as defined in Table 283.
   
   e) Set the CRYPEN bit to 1 in CRYP_CR to start accepting data.
   
   f) Write the B0 packet into CRYP_DIN register, then wait for the CRYPEN bit to be cleared to 0 by the cryptographic processor before moving on to the next phase.

   **Note:** In this init phase data have to be swapped according to the DATATYPE programmed in CRYP_CR register.

2. **CCM header phase**

   The below sequence shall be performed after the CCM Init phase. It must be complete before jumping to the payload phase. The sequence is identical for encryption and decryption. During this phase, the CRYP_DOUT register does not contain any output data.
   
   g) Set the GCM_CCMPH bit to 0b01 in CRYP_CR to indicate that the header phase is ongoing.
   
   h) Set the CRYPEN bit to 1 to start accepting data.
   
   i) If it is the last block of associated data, optionally pad the data with zeros to have a complete block.
   
   j) Append the associated data in the cryptographic processor. The three possible scenarios are described in Section 39.3.5: CRYP procedure to perform a cipher operation.
   
   k) Repeat the previous step until the second last associated data block is processed. For the last block, execute the two previous steps. Once all the additional authenticated data have been supplied, wait until the BUSY flag is cleared.

   **Note:** This phase can be skipped if there is no associated data (Alen=0).

   **Note:** The first block of the associated data B1 must be formatted with the associated data length. This task must be managed by the driver.
3. CCM payload phase (encryption or decryption)
When the payload size is not null, this sequence must be performed after the CCM header phase. During this phase, the encrypted/decrypted payload is stored in the CRYP_DOUT register.

l) Set the CRYPEN bit to 0.
m) Configure GCM_CCMPH bits to 0b10 in CRYP_CR to indicate that the payload phase is ongoing.
n) Select the algorithm direction (0 for encryption, 1 for decryption) through the ALGODIR bit in CRYP_CR.
o) Set the CRYPEN bit to 1 to start accepting data.
p) If it is the last block of cleartext, optionally pad the data with zeros to have a complete block (encryption only). For decryption, refer to Section 39.3.8: CRYP stealing and data padding for more details.
q) Append payload data in the cryptographic processor, and read the result. The three possible scenarios are described in Section 39.3.5: CRYP procedure to perform a cipher operation.
r) Repeat the previous step until the second last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), execute the two previous steps. For the last block of ciphertext (decryption only), the driver must discard the data that is not part of the cleartext when the last block size is less than 16 bytes. Once all payload data have been supplied, wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no payload data, i.e. Plen=0 or Clen=Tlen
Note: Do not forget to remove LSB_{Tlen(C)} encrypted tag information when decrypting ciphertext C.

4. CCM final phase
In this last step, the cryptographic processor generates the CCM authentication tag and stores it in the CRYP_DOUT register.

s) Configure GCM_CCMPH[1:0] bits to 0b11 in CRYP_CR to indicate that the final phase is ongoing and set the ALGODIR bit to 0 in the same register.
t) Load in CRYP_DIN, the CTR0 information which is described in Table 283 with bit[0] set to 0.

Note: In this final phase, data have to be swapped according to the DATATYPE programmed in CRYP_CR register.
u) Wait until the OFNE flag (FIFO output not empty) is set to 1 in the CRYP_SR register.
v) Read the CRYP_DOUT register four times: the output corresponds to the encrypted CCM tag.
w) Disable the cryptographic processor (CRYPEN bit set to 0 in CRYP_CR)
x) If an authenticated decryption is being performed, compare the generated encrypted tag with the encrypted tag padded in the ciphertext, i.e. LSB_{Tlen(C)}= MSB_{Tlen(CRYP_DOUT data)}. 
Suspend/resume operations in CCM mode

Before interrupting the current message in payload phase, the user application must respect the following sequence:

1. If DMA is used, stop the DMA transfers to the IN FIFO by clearing to 0 the DIEN bit in the CRYP_DMACR register.
2. Wait until both the IN and the OUT FIFOs are empty (IFEM=1 and OFNE=0 in the CRYP_SR register) and the BUSY bit is cleared.
3. If DMA is used, stop the DMA transfers from the OUT FIFO by clearing to 0 the DOEN bit in the CRYP_DMACR register.
4. Disable the cryptographic processor by setting the CRYPEN bit to 0 in CRYP_CR, then save the current configuration (bits [9:2], bits [17:16] and bits 19 in the CRYP_CR register). In addition, save the initialization vector registers, since CRYP_IVx registers have changed from their initial values during the data processing.

Note: Key registers do not need to be saved as their original key value is known by the application.
5. Save context swap registers: CRYP_CSGCMCM0..7R
6. If DMA is used, save the DMA controller status (pointers for IN and OUT data transfers, number of remaining bytes, etc.).

To resume message processing, the user application must respect the following sequence:

1. If DMA is used, reconfigure the DMA controller to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Make sure the cryptographic processor is disabled by reading the CRYPEN bit in CRYP_CR (must be 0).
3. Configure the cryptographic processor again with the initial setting in CRYP_CR and key registers using the saved configuration.
4. Restore context swap registers: CRYP_CSGCMCM0..7R
5. Restore CRYP_IVx registers using the saved configuration.
6. Enable the cryptographic processor by setting the CRYPEN bit to 1.
7. If DMA is used, enable again cryptographic processor DMA requests by setting to 1 the DIEN and DOEN bits in the CRYP_DMACR register.

Note: In Header phase DMA OUT FIFO transfer is not used.
39.3.16 CRYP data registers and data swapping

Introduction

The CRYP_DIN register is the 32-bit wide data input register of the peripheral. It is used to enter into the input FIFO up to four 64-bit blocks (TDES) or two 128-bit blocks (AES) of plaintext (when encrypting) or ciphertext (when decrypting), one 32-bit word at a time.

The first word written into the FIFO is the LSB of the input block. The MSB of the input block is written at the end. CRYP_DIN data endianness can be described as below when DATATYPE=“00” (no data swapping):

- In the DES/TDES modes
  Bit 1 (leftmost bit) of the data block corresponds to the MSB (bit 31) of the first word entered into the FIFO, bit 64 (rightmost bit) corresponds to the LSB (bit 0) of the second word entered into the FIFO.

- In the AES mode
  Bit 0 (leftmost bit) of the data block corresponds to the MSB (bit 31) of the first word written into the FIFO, bit 127 (rightmost bit) corresponds to the LSB (bit 0) of the 4th word written into the FIFO.

Similarly CRYP_DOUT register is the 32-bit wide data out register of the peripheral. It is a read-only register that is used to retrieve from the output FIFO up to four 64-bit blocks (TDES) or two 128-bit blocks (AES) of plaintext (when encrypting) or ciphertext (when decrypting), one 32-bit word at a time.

Like for the input data, the LSB of the output block is the first word read from the output FIFO. The MSB of the output block is read at the end. CRYP_DOUT data endianness can be described as below when DATATYPE=“00” (no data swapping):

- In the DES/TDES modes
  Bit 1 (leftmost bit) of the data block corresponds to the MSB (bit 31) of the first word read from the FIFO, bit 64 (rightmost bit) corresponds to the LSB (bit 0) of the second word read from the FIFO.

- In the AES mode
  Bit 0 (leftmost bit) of the data block corresponds to the MSB (bit 31) of the first word read from the FIFO, bit 127 (rightmost bit) corresponds to the LSB (bit 0) of the 4th word read from the FIFO.
DES/TDES data swapping feature

Depending on the type of data to be processed (e.g. byte swapping when data are ASCII text stream), a bit, byte, half-word or no swapping operation must be done on the data read from the input FIFO before entering the little-endian DES processing core. The same swapping must be performed on the data produced by the little-endian DES processing core before they are written to the output FIFO.

*Figure 303* shows how the DES processing core 64-bit data block M1...64 is constructed from two consecutive 32-bit words popped into IN FIFO by the driver. This is done according to the DATATYPE bitfield in the CRYP_CR register.

**Note:** The same swapping is performed between the IN FIFO and the CRYP data block, and between the CRYP data block and the OUT FIFO.

**Figure 303. 64-bit block construction according to the data type (IN FIFO)**

Note: The CRYP Key registers (CRYP_Kx(L/R)) and initialization registers (CRYP_IVx(L/R)) are not sensitive to the swap mode selected. They have a fixed little-endian configuration (refer to *Section 39.3.17* and *Section 39.3.18*, respectively).

A typical example of data swapping is given in *Table 284.*
### Table 284. DES/TDES data swapping feature

<table>
<thead>
<tr>
<th>DATATYPE in CRYP_CR</th>
<th>Swapping performed</th>
<th>Data block representation (64-bit) 0xABCD7720 6973FE01</th>
<th>System memory data (plaintext or cypher)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b00</td>
<td>No swapping</td>
<td>Address @: 0xABCD7720 (LSB, written first) Address @+4: 0x6973FE01</td>
<td>![Swapping Diagram]</td>
</tr>
<tr>
<td>0b01</td>
<td>Half-word (16-bit) swapping</td>
<td>Address @: 0x7720ABCD (swapped LSB, written first) Address @+4: 0x01FE6973</td>
<td>![Swapping Diagram]</td>
</tr>
<tr>
<td>0b10</td>
<td>Byte (8-bit) swapping</td>
<td>Address @: 0x2077CDAB (swapped LSB, written first) Address @+4: 0x01FE7369</td>
<td>![Swapping Diagram]</td>
</tr>
<tr>
<td>0b11</td>
<td>Bit swapping</td>
<td>LSB data word: 0xABCD7720 0b1010 1011 1100 1101 0111 0111 0010 0000 MSB data word: 0x6973FE01 0b0110 1001 0111 0111 1110 0000 0001 Address @: 0x04EEB3D5 (swapped LSB, written first) Address @+4: 0x807FCE96</td>
<td>![Swapping Diagram]</td>
</tr>
</tbody>
</table>
AES data swapping feature

Depending on the type of data to be processed (e.g. byte swapping when data are ASCII text stream), a bit, byte, half-word or no swapping operation must be done on data read from the input FIFO before entering the little-endian AES processing core. The same swapping must be performed on the data produced by the little-endian AES processing core before they are written to the output FIFO.

*Figure 304* shows how the AES processing core 128-bit data block P0..127 is constructed from four consecutive 32-bit words written by the driver to the CRYP_DIN register. This is done according to the DATATYPE bitfield in the CRYP control register (CRYP_CR).

**Note:** The same swapping is performed between the CRYP_DIN and the CRYP data block, and between the CRYP data block and the CRYP_DOUT.

*Figure 304.* 128-bit block construction according to the data type
Note: The swapping operation concerns only the CRYP_DOUT and CRYP_DIN registers. The CRYP_KxL/KxR and CRYP_IVxL/IVxR registers are not sensitive to the swap mode selected. They have a fixed little-endian configuration (refer to Section 39.3.17 and Section 39.3.18).

Typical examples of data swapping are given in Table 285.

| DATATYPE in CRYP_CR | Swapping performed | Data block representation (64-bit)  
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0x4E6F7720 69732074</td>
</tr>
</tbody>
</table>
| 0b00                | No swapping        | Address @: 0x4E6F7720 (LSB, written first)  
|                     |                    | Address @+4: 0x69732074              |
| 0b01                | Half-word (16-bit) swapping | Address @: 0x77204E6F (swapped LSB, written first)  
|                     |                    | Address @+4: 0x20746973              |
| 0b10                | Byte (8-bit) swapping | Address @: 0x20776F4E (swapped LSB, written first)  
|                     |                    | Address @+4: 0x74207369              |
| 0b11                | Bit swapping       | LSB data word: 0x4E6F7720  
|                     |                    | 0b0100 1110 0110 1111 0111 0110 0000  
|                     |                    | MSB data word: 0x69732074  
|                     |                    | 0b0110 1001 0111 0011 0010 0000 0111 0100  
|                     |                    | Address @: 0x4EEF672 (swapped LSB, written first)  
|                     |                    | Address @+4: 0x2E04CE96              |

Table 285. AES data swapping feature

39.3.17 CRYP key registers

The CRYP_Kx registers are used to store the encryption or decryption keys.

They are organized as eight registers in a little-endian configuration, as shown in Table 286.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>k[0:31]</td>
<td>k[32:63]</td>
<td>k[64:95]</td>
<td>k[96:127]</td>
</tr>
</tbody>
</table>

Note: DES/TDES keys include 8-bit parity information that are not used by the cryptographic processor. In other words, bits 8, 16, 24, 32, 40, 48, 56 and 64 of each 64-bit key value Kx[1:64] are not used.
Keys are considered as four 64-bit data items. They therefore do not have the same data format and representation in system memory as plaintext or ciphertext data.

When TDES is selected (ALGOMODE[2:0] = 000 or 001), three key vectors are used (K1, K2 and K3). When DES is selected (ALGOMODE[2:0] = 010 or 011), a single key vector (K1) is used.

Any write operation to the CRYP_Kx(L/R) registers when the BUSY bit is set to 1 in the CRYP_SR register is disregarded (i.e. register content not modified). Thus, the software must check that the BUSY equals 0 before modifying key registers.

Key registers are not affected by the data swapping feature controlled by DATATYPE value in CRYP_CR register.

Refer to Section 39.6: CRYP registers for a detailed description of CRYP_Kx(L/R) registers.

### 39.3.18 CRYP initialization vector registers

The CRYP_IVxL/IVxR registers are used to store the initialization vector or the nonce, depending on the chaining mode selected. When used, these registers are updated by the core after each computation round of the TDES or AES core.

They are organized as four registers in a little-endian configuration, as shown in Table 287.

<table>
<thead>
<tr>
<th>CRYP_IV1R[31:0]</th>
<th>CRYP_IV1L[31:0]</th>
<th>CRYP_IV0R[31:0]</th>
<th>CRYP_IV0L[31:0]</th>
</tr>
</thead>
</table>

Initialization vector registers are considered as two 64-bit data items. They therefore do not have the same data format and representation in system memory as plaintext or ciphertext data.

Any write operation to the CRYP_IV0...1(L/R) registers when the BUSY bit is set to 1 in the CRYP_SR register is disregarded (i.e. register content not modified). Therefore, the software must check that the BUSY equals 0 in the CRYP_SR register before modifying initialization vectors.

Reading the CRYP_IV0...1(L/R) register returns the latest counter value (useful for managing suspend mode) except for CCM/GCM.

*Note:* In DES/TDES mode, only CRYP_IV0x are used.

Initialization vector registers are not affected by the data swapping feature controlled by DATATYPE value in CRYP_CR register.

Refer to Section 39.6: CRYP registers for a detailed description of CRYP_IVxL/IVxR registers.
39.3.19 CRYP DMA interface

The cryptographic processor provides an interface to connect to the DMA (Direct Memory Access) controller. The DMA operation is controlled through the CRYP DMA control register (CRYP_DMACR).

Data input using DMA

DMA can be enabled for writing data into the cryptographic peripheral by setting the DIEN bit in the CRYP_DMACR register. When this bit is set, the cryptographic processor initiates a DMA request during the INPUT phase each time it requires a word to be written to the CRYP_DIN register.

Table 288 shows the recommended configuration to transfer data from memory to cryptographic processor through the DMA controller.

Table 288. Cryptographic processor configuration for memory-to-peripheral DMA transfers

<table>
<thead>
<tr>
<th>DMA channel control register field</th>
<th>Programming recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer size</td>
<td>Message length, multiple of 128-bit. This 128-bit granularity corresponds to two blocks for DES, one block for AES. According to the algorithm and the mode selected, special padding/ciphertext stealing might be required. Refer to Section 39.3.8: CRYP stealing and data padding for details.</td>
</tr>
<tr>
<td>Source burst size (memory)</td>
<td>CRYP FIFO_size /2 /transfer_width = 4</td>
</tr>
<tr>
<td>Destination burst size (peripheral)</td>
<td>CRYP FIFO_size /2 /transfer_width = 4</td>
</tr>
<tr>
<td>(FIFO_size= 8x32-bit, transfer_width= 32-bit)</td>
<td></td>
</tr>
<tr>
<td>DMA FIFO size</td>
<td>CRYP FIFO_size /2 = 16 bytes</td>
</tr>
<tr>
<td>Source transfer width (memory)</td>
<td>32-bit words</td>
</tr>
<tr>
<td>Destination transfer width (peripheral)</td>
<td>32-bit words</td>
</tr>
<tr>
<td>Source address increment (memory)</td>
<td>Yes, after each 32-bit transfer.</td>
</tr>
<tr>
<td>Destination address increment (peripheral)</td>
<td>Fixed address of CRYP_DIN shall be used (no increment).</td>
</tr>
</tbody>
</table>

Data output using DMA

To enable the DMA for reading data from AES peripheral, set the DOEN bit in the CRYP_DMACR register. When this bit is set, the cryptographic processor initiates a DMA request during the OUTPUT phase each time it requires a word to be read from the CRYP_DOUT register.

Table 289 shows the recommended configuration to transfer data from cryptographic processor to memory through the DMA controller.
When AES is used, the cryptographic processor manages two DMA transfer requests through `cryp_in_dma` and `cryp_out_dma` internal input/output signals, which are asserted:
- for IN FIFO: every time a block has been read from FIFO by CRYP,
- for OUT FIFO: every time a block has been written into the FIFO by the cryptographic processor.

When DES is used, the cryptographic processor manages two DMA transfer requests through `cryp_in_dma` and `cryp_out_dma` internal input/output signals, which are asserted:
- for IN FIFO: every time two blocks have been read from FIFO by the cryptographic processor
- for OUT FIFO: every time a word has been written into the FIFO by the cryptographic processor (single transfer). Note that a burst transfer is also triggered when two blocks have been written into the FIFO.

All request signals are deasserted if the cryptographic peripheral is disabled or the DMA enable bit is cleared (DIEN bit for the IN FIFO and DOEN bit for the OUT FIFO in the `CRYP_DMACR` register).

Caution: It is important that DMA controller empties the cryptographic peripheral output FIFO before filling up the CRYP input FIFO. To achieve it, the DMA controller should be configured so
that the transfer from the peripheral to the memory has a higher priority than the transfer from the memory to the peripheral.

For more detailed information on DMA operations, refer to Section 39.3.5: CRYP procedure to perform a cipher operation.

### 39.3.20 CRYP error management

No error flags are generated by the cryptographic processor.

### 39.4 CRYP interrupts

**Overview**

There are two individual maskable interrupt sources generated by the cryptographic processor to signal the following events:

- Input FIFO empty or not full
- Output FIFO full or not empty

These two sources are combined into a single interrupt signal which is the only interrupt signal from the CRYP peripheral that drives the NVIC (nested vectored interrupt controller). You can enable or disable CRYP interrupt sources individually by changing the mask bits in the CRYP.IMSCR register. Setting the appropriate mask bit to 1 enables the interrupt.

The status of the individual maskable interrupt sources can be read either from the CRYP.RISR register, for raw interrupt status, or from the CRYP.MISR register for masked interrupt status. The status of the individual source of event flags can be read from the CRYP.SR register.

*Table 290* gives a summary of the available features.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag in CRYP_xISR(1)</th>
<th>Event flag in CRYP_SR</th>
<th>Enable bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRYP</td>
<td>Output FIFO full</td>
<td>OUTRIS (not masked)</td>
<td>OFFU</td>
<td>OUTIM(2) and CRYPEN</td>
<td>Read one data from output FIFO</td>
</tr>
<tr>
<td></td>
<td>Output FIFO not empty</td>
<td>OUTMIS (masked)</td>
<td>OFNE</td>
<td></td>
<td>Read all data from output FIFO</td>
</tr>
<tr>
<td></td>
<td>Input FIFO not full</td>
<td>INRIS (not masked)</td>
<td>IFNF</td>
<td>INIM(2) and CRYPEN</td>
<td>Write data until input FIFO is full</td>
</tr>
<tr>
<td></td>
<td>Input FIFO empty</td>
<td>INMIS (masked)</td>
<td>IFEM</td>
<td></td>
<td>Write at least one word in input FIFO</td>
</tr>
</tbody>
</table>

1. The flags belong to CRYP.RISR registers (unmasked or raw) or CRYP.MISR registers (masked).
2. The flags belong to CRYP.IMSCR register.
Output FIFO service interrupt - OUTMIS

The output FIFO service interrupt is asserted when there is one or more (32-bit word) data items in the output FIFO. This interrupt is cleared by reading data from the output FIFO until there is no valid (32-bit) word left (that is when the interrupt follows the state of the output FIFO not empty flag OFNE).

The output FIFO service interrupt OUTMIS is NOT enabled with the CRYP enable bit. Consequently, disabling the CRYP will not force the OUTMIS signal low if the output FIFO is not empty.

Input FIFO service interrupt - INMIS

The input FIFO service interrupt is asserted when there are less than four words in the input FIFO. It is cleared by performing write operations to the input FIFO until it holds four or more words.

The input FIFO service interrupt INMIS is enabled with the CRYP enable bit. Consequently, when CRYP is disabled, the INMIS signal is low even if the input FIFO is empty.
39.5 **CRYP processing time**

The time required to process a 128-bit block for each mode of operation is summarized below.

### Table 291. Processing latency for ECB, CBC and CTR

<table>
<thead>
<tr>
<th>Key size</th>
<th>Operating modes</th>
<th>Chaining algorithm</th>
<th>Clock cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 bits</td>
<td>DES encryption or decryption</td>
<td>ECD, CBC</td>
<td>16</td>
</tr>
<tr>
<td>3x64 bits</td>
<td>TDES encryption or decryption</td>
<td>ECD, CBC</td>
<td>48</td>
</tr>
<tr>
<td>128 bits</td>
<td>AES encryption or decryption&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>ECD, CBC, CTR</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>AES key preparation</td>
<td>-</td>
<td>12</td>
</tr>
<tr>
<td>192 bits</td>
<td>AES encryption or decryption&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>ECD, CBC, CTR</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>AES key preparation</td>
<td>-</td>
<td>14</td>
</tr>
<tr>
<td>256 bits</td>
<td>AES encryption or decryption&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>ECD, CBC, CTR</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>AES key preparation</td>
<td>-</td>
<td>16</td>
</tr>
</tbody>
</table>

<sup>(1)</sup> Excluding key preparation time (ECB and CBC only).

### Table 292. Processing time (in clock cycle) for GCM and CCM per 128-bit block

<table>
<thead>
<tr>
<th>Key size</th>
<th>Operating modes</th>
<th>Chaining algorithm</th>
<th>Init phase</th>
<th>Header phase</th>
<th>Payload phase</th>
<th>Tag phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>128 bits</td>
<td>AES encryption or decryption</td>
<td>GCM</td>
<td>24</td>
<td>10</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCM</td>
<td>12</td>
<td>14</td>
<td>25</td>
<td>14</td>
</tr>
<tr>
<td>192 bits</td>
<td>AES encryption or decryption</td>
<td>GCM</td>
<td>28</td>
<td>10</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCM</td>
<td>14</td>
<td>16</td>
<td>29</td>
<td>16</td>
</tr>
<tr>
<td>256 bits</td>
<td>AES encryption or decryption</td>
<td>GCM</td>
<td>32</td>
<td>10</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCM</td>
<td>16</td>
<td>18</td>
<td>33</td>
<td>18</td>
</tr>
</tbody>
</table>
39.6 **CRYP registers**

The cryptographic core is associated with several control and status registers, eight key registers and four initialization vectors registers.

39.6.1 **CRYP control register (CRYP_CR)**

Address offset: 0x00

Reset value: 0x0000 0000

| Bit 31:20 | Reserved, must be kept at reset value. |
| Bit 18 | Reserved, must be kept at reset value. |

Bits 17:16 **GCM_CCMPH[1:0]:** GCM or CCM Phase selection

This bitfield has no effect if GCM, GMAC or CCM algorithm is not selected in ALGOMODE field.

- 00: Init phase
- 01: Header phase
- 10: Payload phase
- 11: Final phase

Bit 15 **CRYPEN: CRYP processor Enable**

0: Cryptographic processor peripheral is disabled

1: Cryptographic processor peripheral is enabled

This bit is automatically cleared by hardware when the key preparation process ends (ALGOMODE= 0b111) or after GCM/GMAC or CCM init phase.

Bit 14 **FFLUSH: CRYP FIFO Flush**

0: No FIFO flush

1: FIFO flush enabled

When CRYPEN = 0, writing this bit to 1 flushes the IN and OUT FIFOs (i.e. read and write pointers of the FIFOs are reset). Writing this bit to 0 has no effect.

When CRYPEN = 1, writing this bit to 0 or 1 has no effect.

Reading this bit always returns 0.

FFLUSH bit has to be set only when BUSY=0. If not, the FIFO is flushed, but the block being processed may be pushed into the output FIFO just after the flush operation, resulting in a non-empty FIFO condition.

Bits 13:10 Reserved, must be kept at reset value.
Bits 9:8 **KEYSIZE[1:0]**: Key Size selection (AES mode only)

This bitfield defines the bit-length of the key used for the AES cryptographic core. This bitfield is ‘don’t care’ in the DES or TDES modes.

00: 128-bit key length
01: 192-bit key length
10: 256-bit key length
11: Reserved, do not use this value

Writing KEYSIZE bits while BUSY=1 has no effect. These bits can only be configured when BUSY=0.

Bits 7:6 **DATATYPE[1:0]**: Data Type selection

This bitfield defines the format of data written in CRYP_DIN or read from CRYP_DOUT registers. For more details refer to Section 39.3.16: CRYP data registers and data swapping).

00: 32-bit data. No swapping for each word. First word pushed into the IN FIFO (or popped off the OUT FIFO) forms bits 1…32 of the data block, the second word forms bits 33…64 etc.
01: 16-bit data, or half-word. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 2 half-words, which are swapped with each other.
10: 8-bit data, or bytes. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 4 bytes, which are swapped with each other.
11: bit data, or bit-string. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 32 bits (1st bit of the string at position 0), which are swapped with each other.

Writing DATATYPE bits while BUSY=1 has no effect. These bits can only be configured when BUSY=0.

Bits 19, 5:3 **ALGOMODE[3:0]**: Algorithm mode

Below definition includes the bit 19:

0000: TDES-ECB (triple-DES Electronic Codebook).
0001: TDES-CBC (triple-DES Cipher Block Chaining).
0010: DES-ECB (simple DES Electronic Codebook).
0011: DES-CBC (simple DES Cipher Block Chaining).
0100: AES-ECB (AES Electronic Codebook).
0101: AES-CBC (AES Cipher Block Chaining).
0110: AES-CTR (AES Counter Mode).
0111: AES key preparation for ECB or CBC decryption.
1000: AES-GCM (Galois Counter Mode) and AES-GMAC (Galois Message Authentication Code mode).
1001: AES-CCM (Counter with CBC-MAC).

Writing ALGOMODE bits while BUSY=1 has no effect. These bits can only be configured when BUSY=0.

Bit 2 **ALGODIR**: Algorithm Direction

0: Encrypt
1: Decrypt

Writing ALGODIR bit while BUSY=1 has no effect. It can only be configured when BUSY=0.

Bits 1:0 Reserved, must be kept at reset value.
39.6.2 CRYP status register (CRYP_SR)

Address offset: 0x04
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>Bit 31-5</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 4</td>
<td>BUSY: Busy bit</td>
</tr>
<tr>
<td>0: The CRYP core is not processing any data. The reason is:</td>
<td></td>
</tr>
<tr>
<td>– either that the CRYP core is disabled (CRYPEN=0 in the CRYP_CR register) and the last processing has completed,</td>
<td></td>
</tr>
<tr>
<td>– or the CRYP core is waiting for enough data in the input FIFO or enough free space in the output FIFO (that is in each case at least 2 words in the DES, 4 words in the AES).</td>
<td></td>
</tr>
<tr>
<td>1: The CRYP core is currently processing a block of data or a key preparation is ongoing (AES ECB or CBC decryption only).</td>
<td></td>
</tr>
<tr>
<td>Bit 3</td>
<td>OFFU: Output FIFO full flag</td>
</tr>
<tr>
<td>0: Output FIFO is not full</td>
<td></td>
</tr>
<tr>
<td>1: Output FIFO is full</td>
<td></td>
</tr>
<tr>
<td>Bit 2</td>
<td>OFNE: Output FIFO not empty flag</td>
</tr>
<tr>
<td>0: Output FIFO is empty</td>
<td></td>
</tr>
<tr>
<td>1: Output FIFO is not empty</td>
<td></td>
</tr>
<tr>
<td>Bit 1</td>
<td>IFNF: Input FIFO not full flag</td>
</tr>
<tr>
<td>0: Input FIFO is full</td>
<td></td>
</tr>
<tr>
<td>1: Input FIFO is not full</td>
<td></td>
</tr>
<tr>
<td>Bit 0</td>
<td>IFEM: Input FIFO empty flag</td>
</tr>
<tr>
<td>0: Input FIFO is not empty</td>
<td></td>
</tr>
<tr>
<td>1: Input FIFO is empty</td>
<td></td>
</tr>
</tbody>
</table>

39.6.3 CRYP data input register (CRYP_DIN)

Address offset: 0x08
Reset value: 0x0000 0000

The CRYP_DIN register is the data input register. It is 32-bit wide. It is used to enter into the input FIFO up to four 64-bit blocks (TDES) or two 128-bit blocks (AES) of plaintext (when encrypting) or ciphertext (when decrypting), one 32-bit word at a time.

To fit different data sizes, the data can be swapped after processing by configuring the DATATYPE bits in the CRYP_CR register. Refer to Section 39.3.16: CRYP data registers and data swapping for more details.
When CRYP_DIN register is written to the data are pushed into the input FIFO.

- If CRYPEN = 1, when at least two 32-bit words in the DES/TDES mode have been pushed into the input FIFO (four words in the AES mode), and when at least two words are free in the output FIFO (four words in the AES mode), the CRYP engine starts an encrypting or decrypting process.

When CRYP_DIN register is read:

- If CRYPEN = 0, the FIFO is popped, and then the data present in the Input FIFO are returned, from the oldest one (first reading) to the newest one (last reading). The IFEM flag must be checked before each read operation to make sure that the FIFO is not empty.
- if CRYPEN = 1, an undefined value is returned.

Note: After the CRYP_DIN register has been read once or several times, the FIFO must be flushed by setting the FFLUSH bit prior to processing new data.

39.6.4 CRYP data output register (CRYP_DOUT)

Address offset: 0x0C
Reset value: 0x0000 0000

The CRYP_DOUT register is the data output register. It is read-only and 32-bit wide. It is used to retrieve from the output FIFO up to four 64-bit blocks (TDES) or two 128-bit blocks (AES) of plaintext (when encrypting) or ciphertext (when decrypting), one 32-bit word at a time.

To fit different data sizes, the data can be swapped after processing by configuring the DATATYPE bits in the CRYP_CR register. Refer to Section 39.3.16: CRYP data registers and data swapping for more details.

When CRYP_DOUT register is read, the last data entered into the output FIFO (pointed to by the read pointer) is returned.

---

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

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>DATAOUT[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>DATAOUT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 DATAOUT[31:0]: Data Input
- On read FIFO is popped (last written value is returned), and its value is returned if CRYPEN=0. If CRYPEN=1 DATAIN register returns an undefined value.
- On write current register content is pushed inside the FIFO.

1380/2974
39.6.5 CRYP DMA control register (CRYP_DMACR)

Address offset: 0x10
Reset value: 0x0000 0000

Bits 31:0 **DATAOUT[31:0]: Data Output**
- On read returns output FIFO content (pointed to by read pointer), else returns an undefined value.
- On write, no effect.

Bit 1 **DOEN:** DMA Output Enable
- When this bit is set, DMA requests are automatically generated by the peripheral during the output data phase.
- 0: DMA for outgoing data transfer is disabled
- 1: DMA for outgoing data transfer is enabled

Bit 0 **DIEN:** DMA Input Enable
- When this bit is set, DMA requests are automatically generated by the peripheral during the input data phase.
- 0: DMA for incoming data transfer is disabled
- 1: DMA for incoming data transfer is enabled

39.6.6 CRYP interrupt mask set/clear register (CRYP_IMSCR)

Address offset: 0x14
Reset value: 0x0000 0000

The CRYP_IMSCR register is the interrupt mask set or clear register. It is a read/write register. When a read operation is performed, this register gives the current value of the mask applied to the relevant interrupt. Writing 1 to the particular bit sets the mask, thus enabling the interrupt to be read. Writing 0 to this bit clears the corresponding mask. All the bits are cleared to 0 when the peripheral is reset.
39.6.7 CRYP raw interrupt status register (CRYP_RISR)

Address offset: 0x18
Reset value: 0x0000 0001

The CRYP_RISR register is the raw interrupt status register. It is a read-only register. When a read operation is performed, this register gives the current raw status of the corresponding interrupt, i.e. the interrupt information without taking CRYP_IMSCR mask into account. Write operations have no effect.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:2</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>OUTIM: Output FIFO service interrupt mask</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Output FIFO service interrupt is masked</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Output FIFO service interrupt is not masked</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>INIM: Input FIFO service interrupt mask</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Input FIFO service interrupt is masked</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Input FIFO service interrupt is not masked</td>
<td></td>
</tr>
</tbody>
</table>

39.6.8 CRYP masked interrupt status register (CRYP_MISR)

Address offset: 0x1C
Reset value: 0x0000 0000

The CRYP_MISR register is the masked interrupt status register. It is a read-only register. When a read operation is performed, this register gives the current masked status of the corresponding interrupt, i.e. the interrupt information taking CRYP_IMSCR mask into account. Write operations have no effect.

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

Bit 1 OUTRIS: Output FIFO service raw interrupt status
This bit gives the output FIFO interrupt information without taking CRYP_IMSCR corresponding mask into account.
0: Raw interrupt not pending
1: Raw interrupt pending

Bit 0 INRIS: Input FIFO service raw interrupt status
This bit gives the input FIFO interrupt information without taking CRYP_IMSCR corresponding mask into account.
0: Raw interrupt not pending
1: Raw interrupt pending
CRYP key registers contain the cryptographic keys.

- In DES/TDES mode, the keys are 64-bit binary values (number from left to right, that is the leftmost bit is bit 1) and named K1, K2 and K3 (K0 is not used). Each key consists of 56 information bits and 8 parity bits.
- In AES mode, the key is considered as a single 128, 192 or 256 bits long sequence \( K_0K_1K_2...K_{127}/191/255 \). The AES key is entered into the registers as follows:
  - for AES-128: \( K_0..K_{127} \) corresponds to \( b_{127}..b_0 \) (\( b_{255}..b_{128} \) are not used),
  - for AES-192: \( K_0..K_{191} \) corresponds to \( b_{191}..b_0 \) (\( b_{255}..b_{192} \) are not used),
  - for AES-256: \( K_0..K_{255} \) corresponds to \( b_{255}..b_0 \).

In all cases key bit \( K_0 \) is the leftmost bit in CRYP inner memory and register bit \( b_0 \) is the rightmost bit in corresponding CRYP_KxLR key register.

For more information refer to Section 39.3.17: CRYP key registers.

Note: Write accesses to these registers are disregarded when the cryptographic processor is busy (bit BUSY = 1 in the CRYP_SR register)

### CRYP key register 0L (CRYP_K0LR)

Address offset: 0x20
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:2</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1</td>
<td>OUTMIS: Output FIFO service masked interrupt status</td>
</tr>
<tr>
<td>Bit 0</td>
<td>INMIS: Input FIFO service masked interrupt status</td>
</tr>
</tbody>
</table>

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

- **Bit 1** OUTMIS: Output FIFO service masked interrupt status
  - This bit gives the output FIFO interrupt information without taking into account the corresponding CRYP.IMSCR mask.
  - 0: Interrupt not pending
  - 1: Interrupt pending

- **Bit 0** INMIS: Input FIFO service masked interrupt status
  - This bit gives the input FIFO interrupt information without taking into account the corresponding CRYP.IMSCR mask.
  - 0: Interrupt not pending
  - 1: Interrupt pending when CRYPEN= 1

### Note
Write accesses to these registers are disregarded when the cryptographic processor is busy (bit BUSY = 1 in the CRYP_SR register)
39.6.10 **CRYP key register 0R (CRYP_K0RR)**

Address offset: 0x24  
Reset value: 0x0000 0000  
Refer to Section 39.6.9: CRYP key register 0L (CRYP_K0LR) for details.

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>K[223:208]</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
</tr>
</tbody>
</table>
| Bits 31:0 | K[223:208]: AES key bit x (x= 255 to 224)  
*Note: This register is not used in DES mode*

39.6.11 **CRYP key register 1L (CRYP_K1LR)**

Address offset: 0x28  
Reset value: 0x0000 0000  
Refer to Section 39.6.9: CRYP key register 0L (CRYP_K0LR) for details.

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>K[191:176]</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
</tr>
</tbody>
</table>
| Bits 31:0 | K[191:176]: AES key bit x (x= 223 to 192)  
*Note: This register is not used in DES mode*

In DES mode, K192 corresponds to key K1 bit 1 and K160 corresponds to key K1 bit 32.
39.6.12 CRYP key register 1R (CRYP_K1RR)

Address offset: 0x2C
Reset value: 0x0000 0000
Refer to Section 39.6.9: CRYP key register 0L (CRYP_K0LR) for details.

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>K[159:144]</strong></td>
</tr>
<tr>
<td>w w w w w w w w w w w w w w w w w w w w w w w</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:0 \textbf{K[159:128]}: AES key bit x (x= 159 to 128)
In DES mode K159 corresponds to key K1 bit 33 and K128 corresponds to key K1 bit 64.

39.6.13 CRYP key register 2L (CRYP_K2LR)

Address offset: 0x30
Reset value: 0x0000 0000
Refer to Section 39.6.9: CRYP key register 0L (CRYP_K0LR) for details.

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>K[127:112]</strong></td>
</tr>
<tr>
<td>w w w w w w w w w w w w w w w w w w w w w w w</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:0 \textbf{K[127:96]}: AES key bit x (x= 127 to 96)
In DES mode K127 corresponds to key K2 bit 1 and K96 corresponds to key K2 bit 32.

39.6.14 CRYP key register 2R (CRYP_K2RR)

Address offset: 0x34
Reset value: 0x0000 0000
Refer to Section 39.6.9: CRYP key register 0L (CRYP_K0LR) for details.

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>K[95:80]</strong></td>
</tr>
<tr>
<td>w w w w w w w w w w w w w w w w w w w w w w w</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>K[79:64]</strong></td>
</tr>
<tr>
<td>w w w w w w w w w w w w w w w w w w w w w w w</td>
</tr>
</tbody>
</table>

ST
### 39.6.15 CRYP key register 3L (CRYP_K3LR)

Address offset: 0x38  
Reset value: 0x0000 0000  
Refer to [Section 39.6.9: CRYP key register 0L (CRYP_K0LR)](#) for details.

<p>| | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>K[63:48]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **K[95:64]**: AES key bit \(x\) (\(x = 95\) to \(64\))  
In DES mode K95 corresponds to key K2 bit 33 and K64 corresponds to key K2 bit 64.

### 39.6.16 CRYP key register 3R (CRYP_K3RR)

Address offset: 0x3C  
Reset value: 0x0000 0000  
Refer to [Section 39.6.9: CRYP key register 0L (CRYP_K0LR)](#) for details.

<p>| | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>K[47:32]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **K[63:32]**: AES key bit \(x\) (\(x = 63\) to \(32\))  
In DES mode K63 corresponds to key K3 bit 1 and K32 corresponds to key K3 bit 32.

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

Bits 31:0 **K[31:0]**: AES key bit \(x\) (\(x = 31\) to \(0\))  
In DES mode K31 corresponds to key K3 bit 33 and K0 corresponds to key K3 bit 64.
### 39.6.17 CRYP initialization vector register 0L (CRYP_IV0LR)

**Address offset:** 0x40  
**Reset value:** 0x0000 0000

The CRYP_IV0...1(L/R)R are the left-word and right-word registers for the initialization vector (64 bits for DES/TDES and 128 bits for AES). For more information refer to [Section 39.3.18: CRYP initialization vector registers](#).

IV0 is the leftmost bit whereas IV63 (DES, TDES) or IV127 (AES) are the rightmost bits of the initialization vector. IV1(L/R)R is used only in the AES. Only CRYP_IV0(L/R) is used in DES/TDES.

**Note:** Write access to these registers are disregarded when the cryptographic processor is busy (bit BUSY = 1 in the CRYP_SR register).

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>IV[0:31]: Initialization vector bit x (x= 0 to 31)</td>
</tr>
</tbody>
</table>

### 39.6.18 CRYP initialization vector register 0R (CRYP_IV0RR)

**Address offset:** 0x44  
**Reset value:** 0x0000 0000

Refer to [Section 39.6.17: CRYP initialization vector register 0L (CRYP_IV0LR)](#) for details.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>IV[32:63]: Initialization vector bit x (x= 32 to 63)</td>
</tr>
</tbody>
</table>
39.6.19  **CRYP initialization vector register 1L (CRYP_IV1LR)**

Address offset: 0x48

Reset value: 0x0000 0000

Refer to Section 39.6.17: CRYP initialization vector register 0L (CRYP_IV0LR) for details.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>IV[64:95]: Initialization vector bit x (x= 64 to 95)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note: This register is not used in DES mode</td>
</tr>
</tbody>
</table>

39.6.20  **CRYP initialization vector register 1R (CRYP_IV1RR)**

Address offset: 0x4C

Reset value: 0x0000 0000

Refer to Section 39.6.17: CRYP initialization vector register 0L (CRYP_IV0LR) for details.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>IV[96:127]: Initialization vector bit x (x= 96 to 127)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note: This register is not used in DES mode</td>
</tr>
</tbody>
</table>

39.6.21  **CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR)**

Address offset: 0x050 + x* 0x4 (x=0 to 7)

Reset value: 0x0000 0000

These registers contain the complete internal register states of the CRYP processor when the GCM/GMAC or CCM algorithm is selected. They are useful when a context swap has to be performed because a high-priority task needs the cryptographic processor while it is already in use by another task.

When such an event occurs, the CRYP_CSGCMCCM0..7R and CRYP_CSGCM0..7R (in GCM/GMAC mode) or CRYP_CSGCMCCM0..7R (in CCM mode) registers have to be read and the values retrieved have to be saved in the system memory space. The cryptographic processor can then be used by the preemptive task. Then when the cryptographic computation is complete, the saved context can be read from memory and written back into the corresponding context swap registers.
### 39.6.22 CRYP context swap GCM registers (CRYP_CSGCMxR)

Address offset: 0x070 + x*0x4 (x=0 to 7)
Reset value: 0x0000 0000

Please refer to Section 39.6.21: CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR) for details.

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

Bits 31:0 **CSGCMCCMx[31:0]**: CRYP processor internal register states for GCM, GMAC and CCM modes.

*Note: This register is not used in DES/TDES or other AES modes than the ones indicated*

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

Bits 31:0 **CSGCMx[31:0]**: CRYP processor internal register states for GCM and GMAC modes.

*Note: This register is not used in DES/TDES or other AES modes than the ones indicated*
## 39.6.23 CRYP register map

**Table 293. CRYP register map and reset values**

| Offset | Register name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x00   | CRYP_CR       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x04   | CRYP_SR       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x08   | CRYP_DIN      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0C   | CRYP_DOUT     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x10   | CRYP_DIMACR   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x14   | CRYP_IMSCR    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x18   | CRYP_RISR     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1C   | CRYP_MISR     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x20   | CRYP_K0LR     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x24   | CRYP_K0RR     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x28   | CRYP_K3LR     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x2C   | CRYP_K3RR     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x30   | CRYP_IV0LR    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x34   | CRYP_IV1LR    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Reset value

| Offset | Register name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|

Reset value

| Offset | Register name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|

Reset value
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>S1</th>
<th>S0</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x44</td>
<td>CRYP_IV0RR</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x48</td>
<td>CRYP_IV1LR</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x4C</td>
<td>CRYP_IV1RR</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x50</td>
<td>CRYP_CSGCMCCM0R</td>
<td></td>
<td></td>
<td>CSGCMCCM0</td>
</tr>
<tr>
<td>0x54</td>
<td>CRYP_CSGCMCCM1R</td>
<td></td>
<td></td>
<td>CSGCMCCM1</td>
</tr>
<tr>
<td>0x58</td>
<td>CRYP_CSGCMCCM2R</td>
<td></td>
<td></td>
<td>CSGCMCCM2</td>
</tr>
<tr>
<td>0x5C</td>
<td>CRYP_CSGCMCCM3R</td>
<td></td>
<td></td>
<td>CSGCMCCM3</td>
</tr>
<tr>
<td>0x60</td>
<td>CRYP_CSGCMCCM4R</td>
<td></td>
<td></td>
<td>CSGCMCCM4</td>
</tr>
<tr>
<td>0x64</td>
<td>CRYP_CSGCMCCM5R</td>
<td></td>
<td></td>
<td>CSGCMCCM5</td>
</tr>
<tr>
<td>0x68</td>
<td>CRYP_CSGCMCCM6R</td>
<td></td>
<td></td>
<td>CSGCMCCM6</td>
</tr>
<tr>
<td>0x6C</td>
<td>CRYP_CSGCMCCM7R</td>
<td></td>
<td></td>
<td>CSGCMCCM7</td>
</tr>
<tr>
<td>0x70</td>
<td>CRYP_CSGCM0R</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x74</td>
<td>CRYP_CSGCM1R</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x78</td>
<td>CRYP_CSGCM2R</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x7C</td>
<td>CRYP_CSGCM3R</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x80</td>
<td>CRYP_CSGCM4R</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 293. CRYP register map and reset values (continued)
Refer to Section 2.3 on page 128 for the register boundary addresses.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x84</td>
<td>CRYP_CSGCM5R</td>
<td>CSGCM5</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x88</td>
<td>CRYP_CSGCM6R</td>
<td>CSGCM6</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x8C</td>
<td>CRYP_CSGCM7R</td>
<td>CSGCM7</td>
<td>00000000000000000000000000000000</td>
</tr>
</tbody>
</table>
40 Hash processor (HASH)

40.1 Introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-224, SHA-256), the MD5 (message-digest algorithm 5) hash algorithm and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of up to \(2^{64}-1\) bits. It also computes 128-bit digests for the MD5 algorithm.

40.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, Secure Hash Standard (SHA-1 and SHA-2 family)
  - Federal Information Processing Standards Publication FIPS PUB 186-4, Digital Signature Standard (DSS)
  - Internet Engineering Task Force (IETF) Request For Comments RFC 1321, MD5 Message-Digest Algorithm
- Fast computation of SHA-1, SHA-224, SHA-256, and MD5
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 66 clock cycles for processing one 512-bit block of data using MD5 algorithm
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Word swapping supported: bits, bytes, half-words and 32-bit words
- Automatic padding to complete the input bit string to fit digest minimum block size of 512 bits (16 x 32 bits)
- Single 32-bit input register associated to an internal input FIFO, corresponding to one block size
- AHB slave peripheral, accessible through 32-bit word accesses only (else an AHB error is generated)
- 8 x 32-bit words (H0 to H7) for output message digest
- Automatic data flow control with support of direct memory access (DMA) using one channel.
- Single or fixed DMA burst transfers of four words
• Interruptible message digest computation, on a per-block basis
  – Re-loadable digest registers
  – Hashing computation suspend/resume mechanism, including DMA

40.3 HASH implementation

The devices have a single instance of HASH peripheral.

40.4 HASH functional description

40.4.1 HASH block diagram

*Figure 305* shows the block diagram of the hash processor.

*Figure 305. HASH block diagram*
40.4.2 HASH internal signals

*Table 294* describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hash_hclk</td>
<td>digital input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>hash_it</td>
<td>digital output</td>
<td>Hash processor global interrupt request</td>
</tr>
<tr>
<td>hash_dma</td>
<td>digital input/output</td>
<td>DMA transfer request/ acknowledge</td>
</tr>
</tbody>
</table>

40.4.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard and the IETF RFC1321 publication (MD5).

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message of any length below $2^{64}$ bits is provided on input, the HASH processing core produces respectively a fixed-length output string called a message digest, defined as follows:

- For MD5 digest size is 128-bit
- For SHA-1 digest size is 160-bit
- For SHA-224 and SHA-256, the digest size is 224 bits and 256 bits, respectively

The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process because the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” by NIST because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest (SHA-1 does not qualify as secure since February 2017). Any change to a message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify.

40.4.4 Message data feeding

The message (or data file) to be processed by the HASH should be considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “01100001 01100010 01100011” is represented by a 32-bit word 0x00636261, and 8-bit words 0x61626300.

Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence HASH_DIN and the FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).
In accordance to the kind of data to be processed (e.g. byte swapping when data are ASCII text stream) there must be a bit, byte, half-word or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core.

**Figure 306** shows how the hash processing core 32-bit data block M0...31 is constructed from one 32-bit words popped into input FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR).

HASH_DIN data endianness when bit swapping is disabled (DATATYPE=00) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

**Figure 306. Message data swapping feature**
40.4.5 Message digest computing

The hash processor sequentially processes several blocks when computing the message digest. For MD5, SHA1 and SHA2, the block size is 512 bits.

Each time the DMA or the CPU writes a block to the hash processor, the HASH automatically starts computing the message digest. This operation is known as partial digest computation.

As described in Section 40.4.4: Message data feeding, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the input FIFO.

In order to perform the hash computation on this data below sequence shall be used by the application:

1. Initialize the hash processor using the HASH_CR register:
   a) Select the right algorithm using ALGO field. If needed program the correct swapping operation on the message input words using DATATYPE bitfield in HASH_CR.
   b) Set MODE=1 and select the key length using LKEY if HMAC mode has been selected.
   c) Update NBLW to define the number of valid bits in last word if it is different from 32 bits. If it is the case automatic padding is applied by the hash processor.

2. Complete the initialization by setting to 1 the INIT bit in HASH_CR. Also set the bit DMAE to 1 if data are transferred via DMA.

Caution: When programming step 2, it is important to set up before or at the same time the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW).

3. Start filling data by writing to HASH_DIN register, unless data are automatically transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the input FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   a) When data are filled by software:
      – The partial digest computation is triggered when the application writes the first word of the next block. Once the processor is ready again (DINIS=1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.
      – The final digest computation is triggered when the last block is entered and the software writes the DCAL bit to 1. If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to writing DCAL bit (see Section 40.4.6 for details).

   b) When data are filled by DMA as a single DMA transfer (MDMAT bit=0):
      – The partial digest computation is triggered automatically each time the FIFO is full.
      – The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register (DCAL bit is set to 1 by hardware). If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to enabling the DMA (see Section 40.4.6 for details).

   c) When data are filled using multiple DMA transfers (MDMAT bit=1) (no MDMA):
      – The partial digest computations are triggered as for single DMA transfers. However the final digest computation is not triggered automatically when the last
block has been transferred to the HASH_DIN register (DCAL bit is not set to 1 by hardware). It allows the hash processor to receive a new DMA transfer as part of this digest computation. To launch the final digest computation, the software must set MDMAT bit to 0 before the last DMA transfer in order to trigger the final digest computation as it is done for single DMA transfers (see description before).

4. Once the digest computation is complete (DCIS= 1), the resulting digest can be read from the output registers as described in Table 295.

Table 295. Hash processor outputs

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Valid output registers</th>
<th>Most significant bit</th>
<th>Digest size (in bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5</td>
<td>HASH_H0 to HASH_H3</td>
<td>HASH_H0[31]</td>
<td>128</td>
</tr>
<tr>
<td>SHA-1</td>
<td>HASH_H0 to HASH_H4</td>
<td>HASH_H0[31]</td>
<td>160</td>
</tr>
<tr>
<td>SHA-224</td>
<td>HASH_H0 to HASH_H6</td>
<td>HASH_H0[31]</td>
<td>224</td>
</tr>
<tr>
<td>SHA-256</td>
<td>HASH_H0 to HASH_H7</td>
<td></td>
<td>256</td>
</tr>
</tbody>
</table>

For more information about HMAC detailed instructions, refer to Section 40.4.7: HMAC operation.

40.4.6 Message padding

Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every block transfer) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Padding processing

Detailed padding sequences with DMA enabled or disabled are described in Section 40.4.5: Message digest computing.

Padding example

As specified by Federal Information Processing Standards PUB 180-4, the message padding consists in appending a “1” followed by $k$ “0”s, itself followed by a 64-bit integer that is equal to the length $L$ in bits of the message. These three padding operations generate a padded message of length $L + 1 + k + 64$, which by construction is a multiple of 512 bits.

For the hash processor, the “1” is added to the last word written to the HASH_DIN register at the bit position defined by the NBLW bitfield, and the remaining upper bits are cleared (“0”s).
Example from FIPS PUB180-4

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length \( L = 24 \):

<table>
<thead>
<tr>
<th>byte 0</th>
<th>byte 1</th>
<th>byte 2</th>
<th>byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>01100001</td>
<td>01100010</td>
<td>01100011</td>
<td>UUUUUUUU</td>
</tr>
</tbody>
</table>

\(--\) 1st word written to HASH_DIN -->

NBLW has to be loaded with the value 24: a “1” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

01100001 01100010 01100011 1UUUUUUU

Since \( L = 24 \), the number of bits in the above bit string is 25, and 423 “0” bits are appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

61626380 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000018

The message length value, \( L \), in two-word format (that is 00000000 00000018) is appended. Hence the final padded message in hexadecimal (byte words in big-endian format):

61626380 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000018

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE=10 in HASH_CR), the above message has to be entered by following the below sequence:

1. 0xUU636261 is written to the HASH_DIN register (where ‘U’ means don’t care).
2. 0x18 is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits).
3. 0x10 is written to the HASH_STR register to start the message padding (described above) and then perform the digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (\( x = 0 \ldots 4 \)) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

\[
\begin{align*}
\text{HASH\_HR0} & = 0xA9993E36 \\
\text{HASH\_HR1} & = 0x4706816A \\
\text{HASH\_HR2} & = 0xBA3E2571 \\
\text{HASH\_HR3} & = 0x7850C26C \\
\text{HASH\_HR4} & = 0x9CD0D89D 
\end{align*}
\]
40.4.7 HMAC operation

Overview

As specified by Internet Engineering Task Force RFC2104 and NIST FIPS PUB 198-1, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

\[
\text{HMAC(message)} = \text{Hash}((\text{Key} | \text{pad}) \text{ XOR opad} | \\
\text{Hash}((\text{Key} | \text{pad}) \text{ XOR ipad} | \text{message}))
\]

where:

- opad = [0x5C]_n (outer pad) and ipad = [0x36]_n (inner pad)
- \([X]_n\) represents a repetition of \(X\) \(n\) times, where \(n\) equal to the size of the underlying hash function data block (\(n = 64\) for 512-bit blocks).
- pad is a sequence of zeroes needed to extend the key to the length \(n\) defined above. If the key length is greater than \(n\), the application shall first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

HMAC processing

Four different steps are required to compute the HMAC:

1. The block is initialized by writing the INIT bit to 1 with the MODE bit at 1 and the ALGO bits set to the value corresponding to the desired algorithm. The LKEY bit must also be set to 1 if the key being used is longer than 64 bytes. In this case, as required by HMAC specifications, the hash processor will use the hash of the key instead of the real key.

2. The key to be used for the inner hash function must be provided to the hash processor. The key loading operation follows the same mechanism as the message bit string loading, i.e. write key data into HASH_DIN and complete the transfer by writing to HASH_STR register.

Note: Endianness details can be found in Section 40.4.4: Message data feeding.

3. Once the last key word has been entered and computation has started, the hash processor elaborates the inner key material. Once this operation has completed, it is ready to accept the message bit string as described in Section 40.4.4: Message data feeding.

4. After the final hash round, the hash processor returns “ready” to indicate that it is ready to receive the key to be used for the outer hash function (normally, this key is the same as the one used for the inner hash function). When the last word of the key is entered and the computation has started, the HMAC result can be found in the HASH_HR0 to HASH_HR7 registers as soon as DCIS = 1.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 40.6: HASH processing time.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO=00 and MODE=1 in HASH_CR) as specified by NIST.

Let us assume that the original message is the ASCII binary-coded form of "Sample message for keylen=blocklen", of length \(L = 34\) bytes. If the HASH is programmed in
no swapping mode (DATATYPE=00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. **Inner hash key** input (length=64, i.e. no padding), specified by NIST. As key length=64, LKEY bit is set to 0 in HASH_CR register
   
   00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617
   18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F
   30313233 34353637 38393A3B 3C3D3E3F

2. **Message** input (length=34, i.e. padding required). HASH_STR must be set to 0x20 to start message padding and inner hash computation (see 'U' as don’t care)
   
   53616D70 6C65206D 65737361 67652066 6F72206B 65796C65
   6E3D626C 6F636B6C 656EUUUU

3. **Outer hash key** input (length=64, i.e. no padding). A key identical to the inner hash key is entered here.

4. **Final outer hash computing** is then performed by the HASH. The HMAC-SHA1 digest result is available in the HASH_HRx registers (x = 0 to 4), as shown below:
   
   HASH_HR0 = 0x5FD596EE
   HASH_HR1 = 0x78D5553C
   HASH_HR2 = 0x8FF4E72D
   HASH_HR3 = 0x266DFD19
   HASH_HR4 = 0x2366DA29
40.4.8 HASH suspend/resume operations

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in Figure 307.

Figure 307. HASH suspend/resume mechanism

To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described below.
Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing 16 words 32-bit (plus one if it is the first block):

1. In Polling mode, wait for BUSY = 0, then poll if the DINIS status bit is set to 1.
   In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37. HASH_CSR38 to HASH_CSR53 registers must also be saved if an HMAC operation was ongoing.

To resume the processing of a message, proceed as follows:

1. Write the following registers with the values saved in memory: HASH_IMR, HASH_STR and HASH_CR.
2. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point where it has been interrupted.

Note: To optimize the resume process when NBW[3:0]=0x0, HASH_CSR22 to HASH_CSR37 registers do not need to be saved then restored as the FIFO is empty.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is recommended to suspend and then restore a secure digest computing is described below.

To suspend the processing of a message using DMA, proceed as follows:

1. In Polling mode, wait for BUSY = 0. If DCIS is set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 2.
2. In Polling mode, wait for BUSY = 1.
3. Disable the DMA channel. Then clear DMAE bit in HASH_CR register.
4. In Polling mode, wait for BUSY = 0. If DCIS is set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 5.
5. Save HASH_IMR, HASH_STR, HASH_CR, and HASH_CSR0 to HASH_CSR37 registers. HASH_CSR38 to HASH_CSR53 registers must also be saved if an HMAC operation was ongoing.
To resume the processing of a message using DMA, proceed as follows:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again. Do not forget to take into account the words that have already pushed into the FIFO if NBW[3:0] is higher than 0x0.
2. Program the values saved in memory to HASH_IMR, HASH_STR and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point where it was interrupted by setting the DMAE bit.

**Note:** To optimize the resume process when NBW[3:0]=0x0, HASH_CSR22 to HASH_CSR37 registers do not need to be saved then restored as the FIFO is empty.

### 40.4.9 HASH DMA interface

The HASH supports both single and fixed DMA burst transfers of four words.

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH initiates a DMA request each time a block has to be written to the HASH_DIN register.

Once four 32-bit words have been received, the HASH automatically triggers a new request to the DMA. For more information refer to Section 40.4.5: Message digest computing.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that will be copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

\[ \text{NBLW} = \text{Len}(\text{Message}) \mod 32 \]  

where “\( x \mod 32 \)” gives the remainder of \( x \) divided by 32.

The DMAS bit of the HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared and no DMA transfer is ongoing.

**Note:** No interrupt is associated to DMAS bit. 
When MDMAT is set, the size of the transfer must be a multiple of four words.

### 40.4.10 HASH error management

No error flags are generated by the hash processor.

### 40.5 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal the following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal (hash_it), which is in turn connected to the NVIC (nested vectored interrupt controller). Each interrupt source can individually be enabled or disabled by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit to 1 enables the interrupt.
The status of each maskable interrupt source can be read from the HASH_SR register. Table 296 gives a summary of the available features.

### Table 296. HASH interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>HASH</td>
<td>Digest computation completed</td>
<td>DCIS</td>
<td>DCIE</td>
<td>Clear DCIS or set INIT</td>
</tr>
<tr>
<td></td>
<td>Data input buffer ready to get a new block</td>
<td>DINIS</td>
<td>DINIE</td>
<td>Clear DINIS or write to HASH_DIN</td>
</tr>
</tbody>
</table>

#### 40.6 HASH processing time

Table 297 summarizes the time required to process an intermediate block for each mode of operation.

### Table 297. Processing time (in clock cycle)

<table>
<thead>
<tr>
<th>Mode of operation</th>
<th>FIFO load(1)</th>
<th>Computation phase</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5</td>
<td>16</td>
<td>50</td>
<td>66</td>
</tr>
<tr>
<td>SHA-1</td>
<td>16</td>
<td>66</td>
<td>82</td>
</tr>
<tr>
<td>SHA-224</td>
<td>16</td>
<td>50</td>
<td>66</td>
</tr>
<tr>
<td>SHA-256</td>
<td>16</td>
<td>66</td>
<td></td>
</tr>
</tbody>
</table>

1. Add the time required to load the block into the processor.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- **1 to 2.5** for a hash message
- **~2.5** for an HMAC input-key
- **1 to 2.5** for an HMAC message
- **~2.5** for an HMAC output key in case of a short key
- **3.5 to 5** for an HMAC output key in case of a long key
40.7 HASH registers

The HASH core is associated with several control and status registers and several message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB error is generated.

40.7.1 HASH control register (HASH_CR)

Address offset: 0x00
Reset value: 0x0000 0000

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

Bits 31:19 Reserved, must be kept at reset value.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **LKEY**: Long key selection

This bit selects between short key (≤ 64 bytes) or long key (> 64 bytes) in HMAC mode.

0: Short key (≤ 64 bytes)
1: Long key (> 64 bytes)

This selection is only taken into account when the INIT bit is set and MODE = 1.

Changing this bit during a computation has no effect.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **MDMAT**: Multiple DMA transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.
1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE**: DIN not empty

This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is written to 1.

0: No data are present in the data input buffer
1: The input buffer contains at least one word of data

This bit is read-only.
Bits 11:8 **NBW[3:0]**: Number of words already pushed

This bitfield reflects the number of words in the message that have already been pushed into the IN FIFO. NBW is incremented by one when a write access to the HASH_DIN register is performed (except if DINNE = 0 and the DMA is not used, see below description). NBW goes to zero when the INIT bit is written to 1. This bitfield is read-only.

**If the DMA is not used**

0000: if DINNE = 0, no word has been pushed into the DIN buffer (both HASH_DIN register and IN FIFO are empty), otherwise one word has been pushed into the DIN buffer (HASH_DIN register contains one word and IN FIFO is empty)

0001: two words have been pushed into the DIN buffer (i.e. HASH_DIN register and the IN FIFO contain one word each)

... 

1111: 16 words have been pushed into the DIN buffer.

**If the DMA is used**

NBW contains the exact number of words that have been pushed into the IN FIFO by the DMA.

Bits 18, 7 **ALGO[1:0]**: Algorithm selection

These bits select the hash algorithm.

- 00: SHA-1
- 01: MD5
- 10: SHA-224
- 11: SHA-256

This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect.

Bit 6 **MODE**: Mode selection

This bit selects the HASH or HMAC mode for the selected algorithm:

- 0: Hash mode selected
- 1: HMAC mode selected. LKEY must be set if the key being used is longer than 64 bytes.

This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

Bits 5:4 **DATATYPE[1:0]**: Data type selection

Defines the format of the data entered into the HASH_DIN register:

- 00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.
- 01: 16-bit data, or half-word. The data written into HASH_DIN are considered as two half-words, and are swapped before being used by the HASH processing.
- 10: 8-bit data, or bytes. The data written into HASH_DIN are considered as four bytes, and are swapped before being used by the HASH processing.
- 11: bit data, or bit-string. The data written into HASH_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).
Bit 3 **DMAE**: DMA enable
   0: DMA transfers disabled
   1: DMA transfers enabled. A DMA request is sent as soon as the HASH core is ready to receive data.
   After this bit is set it is cleared by hardware while the last data of the message is written into the hash processor.
   Setting this bit to 0 while a DMA transfer is on-going is not aborting this current transfer. Instead, the DMA interface of the IP remains internally enabled until the transfer is completed or INIT is written to 1.
   Setting INIT bit to 1 does not clear DMAE bit.

Bit 2 **INIT**: Initialize message digest calculation
   Writing this bit to 1 resets the hash processor core, so that the HASH is ready to compute the message digest of a new message.
   Writing this bit to 0 has no effect. Reading this bit always return 0.

Bits 1:0 Reserved, must be kept at reset value.
40.7.2 **HASH data input register (HASH_DIN)**

Address offset: 0x04  
Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks. When the HASH_DIN register is programmed, the value presented on the AHB databus is 'pushed' into the hash core and the register takes the new value presented on the AHB databus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.

When a complete block has been written to the HASH_DIN register, an intermediate digest calculation is launched:
- by writing new data into the HASH_DIN register (the first word of the next block) if the DMA is not used (intermediate digest calculation),
- automatically if the DMA is used.

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched by writing the DCAL bit to 1 in the HASH_STR register (final digest calculation). This operation is automatic if the DMA is used and MDMAT bit is set to 0.

Reading the HASH_DIN register returns the last word written to this location (zero after reset).

*Note:* When the HASH is busy, a write access to the HASH_DIN register might stall the AHB bus if the digest calculation (intermediate or final) is not complete.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>DATAIN[31:0]: Data input</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Writing this register pushes the current register content into the IN FIFO, and the register takes the new value presented on the AHB databus.</td>
</tr>
<tr>
<td></td>
<td>Reading this register returns the current register content.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>DATAIN[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
40.7.3 HASH start register (HASH_STR)

Address offset: 0x08
Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register)
- It is used to start the processing of the last block in the message by writing the DCAL bit to 1

| Bits 31:9 | Reserved, must be kept at reset value. |
| Bit 8   | DCAL: Digest calculation |
| Writing this bit to 1 starts the message padding, using the previously written value of NBLW, and starts the calculation of the final message digest with all data words written to the input FIFO since the INIT bit was last written to 1. |
| Reading this bit returns 0. |

| Bits 7:5 | Reserved, must be kept at reset value. |
| Bits 4:0 | NBLW[4:0]: Number of valid bits in the last word |
| When the last word of the message bit string is written in HASH_DIN register, the hash processor takes only the valid bits specified as below, after internal data swapping: |
| 0x00: All 32 bits of the last data written are valid message bits i.e. M[31:0] |
| 0x01: Only one bit of the last data written (after swapping) is valid i.e. M[0] |
| 0x02: Only two bits of the last data written (after swapping) are valid i.e. M[1:0] |
| 0x03: Only three bits of the last data written (after swapping) are valid i.e. M[2:0] |
| ... |
| 0x1F: Only 31 bits of the last data written (after swapping) are valid i.e. M[30:0] |

The above mechanism is valid only if DCAL=0. If NBLW bits are written while DCAL is set to 1, the NBLW bitfield remains unchanged. In other words it is not possible to configure NBLW and set DCAL at the same time.

Reading NBLW bits returns the last value written to NBLW.
40.7.4 HASH digest registers

These registers contain the message digest result named as follows:
- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result.
- HASH_HR0, HASH_HR1, HASH_HR2 and HASH_HR3 registers return A, B, C and D (respectively), as defined by MD5.
- HASH_HR0 to HASH_HR6 registers return the SHA-224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA-256 digest result.

In all cases, the digest most significant bit is stored in HASH_H0[31] and unused HASH_HRx registers are read as zeros.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (DCIS bit equals 0), then the read operation is stalled until the hash calculation has completed.

Note: When starting a digest computation for a new message (by writing the INIT bit to 1), HASH_HRx registers are forced to their reset values.

HASH_HR0 to HASH_HR4 registers can be accessed through two different addresses.

**HASH aliased digest register x (HASH_HRAx)**

Address offset: 0x0C + 0x04 * x (x=0 to 4)
Reset value: 0x0000 0000

The content of the HASH_HRAx registers is identical to the one of the HASH_HRx registers located at address offset 0x310.

**HASH digest register x (HASH_HRx)**

Address offset: 0x310 + 0x04 * x (x=0 to 4)
Reset value: 0x0000 0000

Bits 31:0 Hx[31:0]: Hash data x
Refer to Section 40.7.4: HASH digest registers introduction.
Bits 31:0  **Hx[31:0]**: Hash data x
Refer to Section 40.7.4: HASH digest registers introduction.

**HASH supplementary digest register x (HASH_HRx)**
Address offset: 0x310 + 0x04 * x (x = 5 to 7)
Reset value: 0x0000 0000

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

Bits 31:0  **Hx[31:0]**: Hash data x
Refer to Section 40.7.4: HASH digest registers introduction.

### 40.7.5  HASH interrupt enable register (HASH_IMR)
Address offset: 0x20
Reset value: 0x0000 0000

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

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

- **Bit 1**  **DCIE**: Digest calculation completion interrupt enable
  0: Digest calculation completion interrupt disabled
  1: Digest calculation completion interrupt enabled.

- **Bit 0**  **DINIE**: Data input interrupt enable
  0: Data input interrupt disabled
  1: Data input interrupt enabled
40.7.6 HASH status register (HASH_SR)

Address offset: 0x24
Reset value: 0x0000 0001

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

Bit 3 BUSY: Busy bit
0: No block is currently being processed
1: The hash core is processing a block of data

Bit 2 DMAS: DMA Status
This bit provides information on the DMA interface activity. It is set with DMAE=0 and cleared when DMAE=0 and no DMA transfer is ongoing. No interrupt is associated with this bit.
0: DMA interface is disabled (DMAE=0) and no transfer is ongoing
1: DMA interface is enabled (DMAE=1) or a transfer is ongoing

Bit 1 DCIS: Digest calculation completion interrupt status
This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by writing the INIT bit to 1 in the HASH_CR register.
0: No digest available in the HASH_HRx registers (zeros are returned)
1: Digest calculation complete, a digest is available in the HASH_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 DINIS: Data input interrupt status
This bit is set by hardware when the FIFO is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.
0: Less than 16 locations are free in the input buffer
1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.
When DINIS=0, HASH_CSRx registers read as zero.

40.7.7 HASH context swap registers

These registers contain the complete internal register states of the hash processor. They are useful when a suspend/resume operation has to be performed because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task, and when the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSRx registers can be read only when DINIS equals to 1, otherwise zeros are returned.
**HASH context swap register 0 (HASH_CSR0)**

Address offset: 0x0F8  
Reset value: 0x0000 0002

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

**HASH context swap register x (HASH_CSRx)**

Address offset: 0x0F8 + 0x4 * x (x = 1 to 53)  
Reset value: 0x0000 0000

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

---

**Note:** The tables represent the bit fields of the context swap registers, indicating their read/write accessibility and default values.
### 40.7.8 HASH register map

Table 298 gives the summary HASH register map and reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register size</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>HASH_CR</td>
<td>ALGO[1]</td>
<td>0000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LKEY</td>
<td>0000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MDMA[1]</td>
<td>0000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DINNE</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x04</td>
<td>HASH_DIN</td>
<td>DATAIN[31:16]</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x08</td>
<td>HASH_STR</td>
<td>ALGO[0]</td>
<td>0000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LKEY</td>
<td>0000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MDMA[0]</td>
<td>0000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DINNE</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x0C</td>
<td>HASH_HRAU</td>
<td>H0</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x10</td>
<td>HASH_HRA1</td>
<td>H1</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x14</td>
<td>HASH_HRA2</td>
<td>H2</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x18</td>
<td>HASH_HRA3</td>
<td>H3</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x1C</td>
<td>HASH_HRA4</td>
<td>H4</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x20</td>
<td>HASH_IMR</td>
<td>H5</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x24</td>
<td>HASH_SR</td>
<td>H6</td>
<td>0000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BUSY</td>
<td>0000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DMAE</td>
<td>0000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INIT</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x28</td>
<td>HASH_CSR0</td>
<td>CSR0</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x2C</td>
<td>HASH_CSR1</td>
<td>CSR1</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x30</td>
<td>HASH_CSR2</td>
<td>CSR2</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x34</td>
<td>HASH_CSR3</td>
<td>CSR3</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x38</td>
<td>HASH_CSR4</td>
<td>CSR4</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x3C</td>
<td>HASH_CSR5</td>
<td>CSR5</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x40</td>
<td>HASH_CSR6</td>
<td>CSR6</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x44</td>
<td>HASH_CSR7</td>
<td>CSR7</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x48</td>
<td>HASH_CSR8</td>
<td>CSR8</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x4C</td>
<td>HASH_CSR9</td>
<td>CSR9</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x50</td>
<td>HASH_CSR10</td>
<td>CSR10</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x54</td>
<td>HASH_CSR11</td>
<td>CSR11</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x58</td>
<td>HASH_CSR12</td>
<td>CSR12</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x5C</td>
<td>HASH_CSR13</td>
<td>CSR13</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x60</td>
<td>HASH_CSR14</td>
<td>CSR14</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x64</td>
<td>HASH_CSR15</td>
<td>CSR15</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x68</td>
<td>HASH_CSR16</td>
<td>CSR16</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x6C</td>
<td>HASH_CSR17</td>
<td>CSR17</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x70</td>
<td>HASH_CSR18</td>
<td>CSR18</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x74</td>
<td>HASH_CSR19</td>
<td>CSR19</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x78</td>
<td>HASH_CSR20</td>
<td>CSR20</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x7C</td>
<td>HASH_CSR21</td>
<td>CSR21</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x80</td>
<td>HASH_CSR22</td>
<td>CSR22</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x84</td>
<td>HASH_CSR23</td>
<td>CSR23</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x88</td>
<td>HASH_CSR24</td>
<td>CSR24</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x8C</td>
<td>HASH_CSR25</td>
<td>CSR25</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x90</td>
<td>HASH_CSR26</td>
<td>CSR26</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x94</td>
<td>HASH_CSR27</td>
<td>CSR27</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x98</td>
<td>HASH_CSR28</td>
<td>CSR28</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x9C</td>
<td>HASH_CSR29</td>
<td>CSR29</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xA0</td>
<td>HASH_CSR30</td>
<td>CSR30</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xA4</td>
<td>HASH_CSR31</td>
<td>CSR31</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xA8</td>
<td>HASH_CSR32</td>
<td>CSR32</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xAC</td>
<td>HASH_CSR33</td>
<td>CSR33</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xB0</td>
<td>HASH_CSR34</td>
<td>CSR34</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xB4</td>
<td>HASH_CSR35</td>
<td>CSR35</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xB8</td>
<td>HASH_CSR36</td>
<td>CSR36</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xBC</td>
<td>HASH_CSR37</td>
<td>CSR37</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xC0</td>
<td>HASH_CSR38</td>
<td>CSR38</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xC4</td>
<td>HASH_CSR39</td>
<td>CSR39</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xC8</td>
<td>HASH_CSR40</td>
<td>CSR40</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xCC</td>
<td>HASH_CSR41</td>
<td>CSR41</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xD0</td>
<td>HASH_CSR42</td>
<td>CSR42</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xD4</td>
<td>HASH_CSR43</td>
<td>CSR43</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xD8</td>
<td>HASH_CSR44</td>
<td>CSR44</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xDC</td>
<td>HASH_CSR45</td>
<td>CSR45</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xE0</td>
<td>HASH_CSR46</td>
<td>CSR46</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xE4</td>
<td>HASH_CSR47</td>
<td>CSR47</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xE8</td>
<td>HASH_CSR48</td>
<td>CSR48</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xEC</td>
<td>HASH_CSR49</td>
<td>CSR49</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xF0</td>
<td>HASH_CSR50</td>
<td>CSR50</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xF4</td>
<td>HASH_CSR51</td>
<td>CSR51</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xF8</td>
<td>HASH_CSR52</td>
<td>CSR52</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0xFC</td>
<td>HASH_CSR53</td>
<td>CSR53</td>
<td>0000000000000000</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
41 On-The-Fly Decryption engine - AXI (OTFDEC)

41.1 Introduction

On-the-fly decryption engine OTFDEC allows to decrypt on-the-fly AXI traffic based on the read request address information. Four independent and non-overlapping encrypted regions can be defined in OTFDEC, with optional execute-only or execute-never enforcement per region.

OTFDEC is using AES-128 in counter mode to achieve the lowest possible latency. As a consequence, each time the content of one encrypted region is changed the entire region must be re-encrypted with a different cryptographic context (key or initialization vector). This constraint makes OTFDEC suitable to decrypt read-only data or code, stored for example in external NOR Flash memory.

Note: When OTFDEC is used in conjunction with OCTOSPI it is mandatory to access the external read-only memory using the memory map mode of the Flash memory controller.

CPU memories and OTFDEC follow little endian notation whereas AES hardware accelerator follows big endian notation. See AN5281 “How to use OTFDEC for encryption/decryption in trusted environment on STM32 MCUs” for more details.

41.2 OTFDEC main features

- On-the-fly 128-bit decryption during STM32 OCTOSPI memory-mapped read operations (single or multiple).
  - Use of AES in counter (CTR) mode, with keystream FIFO (depth=4)
  - Support for any read size
  - Physical address of the reads is used for the encryption/decryption
- Up to four independent encrypted regions
  - Granularity of the region definition: 4096 bytes
  - Region configuration write locking mechanism
  - Two optional decryption modes: execute-only and execute-never.
  - Each region has its own 128-bit key, two bytes firmware version, and eight bytes application-defined nonce. At least one of those must be changed each time an encryption is performed by the application.
- Encryption keys confidentiality and integrity protection
  - Write-only registers, with software locking mechanism
  - Availability of 8-bit CRC as public key information
- Support for STM32 OctoSPI pre-fetching mechanism.
- In execute-only mode, possibility to select an enhanced encryption mode to add a proprietary layer of protection on top of AES stream cipher.
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored).
- Encryption mode
41.3 OTFDEC functional description

41.3.1 OTFDEC block diagram

*Figure 308* shows the block diagram of the OTFDEC.

*Figure 308. OTFDEC block diagram*
41.3.2 OTFDEC internal signals

Table 299 describes a list of useful to know internal signals available at OTFDEC level, not at the product level (on pads).

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>otfdec_hclk</td>
<td>digital input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>otfdec_it</td>
<td>digital output</td>
<td>OTFDEC global interrupt request</td>
</tr>
</tbody>
</table>

41.3.3 OTFDEC on-the-fly decryption

Introduction

Typical usage for OTFDEC is shown on Figure 309.

Figure 309. Typical OTFDEC usage in the device

Original purpose of OTFDEC is to protect the confidentiality of execute-only firmware libraries, executed from external SPI NOR Flash memory devices. This scheme is part of
STM32 family of proprietary code read-out protection, leveraging on the capability to securely load secrets during microcontroller start up. OTFDEC protection also applies to read-only "code + data libraries", and to execute-never, read-only data stored in external memory.

**Note:** OTFDEC provides optional execute-only and execute-never enforcement on any of the four encrypted regions.

A special locking scheme is available in OTFDEC in order to protect the integrity of the decryption keys and also to protect the other configurations against software denial of services attacks.

When OTFDEC is used in conjunction with OCTOSPI it is mandatory to access the external read-only memory using the memory map mode of the Flash memory controller.

**OTFDEC architecture**

OTFDEC principle is to analyze all read address channel transactions on the AXI interconnect between the host and a target, like the OCTOSPI controller shown on Figure 309.

If the read request is within one of the four regions programmed in OTFDEC the control logic will trigger a keystream computation based on AES algorithm in counter mode. This keystream is then used to decrypt on-the-fly the data present in the read response channel, tying low the RREADY signal while the keystream information is being computed (this takes up to 11 cycles). Any accesses outside the enabled OTFDEC regions belong to a non-encrypted region.

Each OTFDEC regions are programmed through registers RxCFGR, RxSTARTADDR, RxENDADDR, RxNONCER and RxKEYR, where x=1 to 4. In register RxCFGR the MODE bits define whether the region is code (execute-only), data (execute-never), or both.

Granularity for the region determination is 4096 bytes.

**Note:** Although OTFDEC does not prevent region overlapping it is not a valid programming and it should be avoided by application software.

OTFDEC can decrypt incremental or wrap bursts only if they do not cross the 4096-byte aligned address boundaries.

**41.3.4 AES in counter mode decryption**

*Figure 310* shows how OTFDEC uses industry standard Advanced Encryption Standard (AES) algorithm in counter chaining mode. This mode is specified by NIST in *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*. 
Every 128-bit data block a special keystream information is computed using AES block cipher, as defined below:

- The initialization vector $AES\_IV[127:0] = R\times NONCER[63:0] \ || \ 0b0000\ 0000\ 0000\ 0000\ || \ R\times CFGR[31:16] \ || \ 0b00 \ || \ (x-1) \ || \ ReadAddress[31:4]$ (address modulo 128-bit)
- The key material $AES\_KEY[127:0] = R\times KEYR3[31:0] \ || \ R\times KEYR2[31:0] \ || \ R\times KEYR1[31:0] \ || \ R\times KEYR0[31:0]$

`Note:` Above $x$ is the RegionID of the selected encrypted region ($x=1$ to $4$).

Resulting 128-bit keystream is XORed with 128-bit cipher text data to produce the 128-bit clear text data.
- $AES\_DIN$ and $AES\_DOUT$ data blocks are constructed following the rule below:
  $AES\_Dx[127:0] = AXI\_word(@+0x8)[63:0] \ || \ AXI\_word(@)[63:0]$, where $@$ is the hexadecimal address used to compute the keystream.

When the read request is not within an encrypted region, or the decryption is not enabled in this region the 128-bit AXI data is not changed.

`Note:` When application sets the MODE bitfield to 11 in OTFDEC_RxCFGR an additional layer of protection is added on top of the AES stream cipher. This enhanced encryption mode is valid only in execute-only.
41.3.5 Flow control management

Figure 311 shows how OTFDEC manages two back-to-back AXI requests for a burst read of 4 words (256-bit). All 128-bit data blocks have contiguous address values.

Figure 311. OTFDEC flow control overview (dual burst read request)

Few notes on this diagram:

1. OTFDEC enforces RREADY low as it is not ready to decrypt data (keystream computation).
2. OTFDEC does not enforce RREADY low as a valid mask is ready to XOR the incoming data. The decryption is done with zero latency, as expected.
3. The keystream FIFO is full. Next time a keystream is needed 11 clock cycles will be needed.
4. STM32 OctoSPI controller has a pre-fetching mechanism that greatly speed up any read request to an address that is consecutive to the last one. OTFDEC is able to manage this shorter latency thanks to the mask FIFO mechanism.
Figure 312 shows how OTFDEC manages an AXI request for a burst read of 4 words (256-bit), followed by a single read on an address that is not contiguous to the previous one.

Figure 312. OTFDEC flow control overview (burst then single read request)

Few notes on this diagram:
1. OTFDEC enforces RREADY low as it is not ready to decrypt data.
2. OTFDEC enforces RREADY low as pre-computed mask cannot be used for this request (the encrypted region address is not contiguous).
41.3.6 OTFDEC error management

OTFDEC automatically manages errors defined as below:

- Illegal read to OTFDEC_RxKEYR registers
- Illegal write to OTFDEC_RxKEYR registers while CONFIGLOCK or KEYLOCK="1" in OTFDEC_RxCFGR.
- Illegal write to OTFDEC_RxCFGR, OTFDEC_RxSTARTADD, OTFDEC_RxENDADD or OTFDEC_RxNONCER registers while CONFIGLOCK="1" in OTFDEC_RxCFGR (x=1 to 4).
- Illegal read to an execute-only region (MODE[1:0]=00 or 11), and illegal execution request to an execute-never region (MODE[1:0]=01). Such illegal requests return 0x0, without bus error.
- Key error: read requests to a encrypted region after the key registers have been cleared by an abort event (tamper detection, unauthorized debug connection, untrusted boot, RDP level regression). Such read requests return 0x0, without bus error.

An interrupt can be generated by one or more of above errors if the bit SEIE, XONEIE or KEIE is set in OTFDEC_IER register, as explained in next Section 41.4.

Note: After a key error OTFDEC keys must be initialized again, and a reset of OTFDEC might be needed if registers are locked.

41.4 OTFDEC interrupts

There are three independent maskable interrupt sources generated by the OTFDEC, signaling following security events:

- Illegal read or write access to keys (SEIF flag), see Section 41.3.6
- Illegal write to a region’s configuration while CONFIGLOCK=1 (SEIF flag), see Section 41.3.6
- Read access to an execute-only region (MODE[1:0]=00 or 11), execute access to a execute-never data region (MODE[1:0]=01). Both are triggering the XONEIF flag.
- Key error (encrypted regions read as zero), triggering the KEIF flag. See Section 41.3.6.

Interrupt sources are connected to the same global interrupt request signal.

You can enable or disable OTFDEC interrupt sources by setting corresponding SEIE, XONEIE or KEIE bits in OTFDEC_IER register, as described in Table 300. Status of the interrupt event is found in OTFDEC_ISR register, and this event can be cleared using OTFDEC_ICR register.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Error</td>
<td>SEIF</td>
<td>SEIE</td>
</tr>
<tr>
<td>Execute-only, Execute-Never Error</td>
<td>XONEIF</td>
<td>XONEIE</td>
</tr>
<tr>
<td>Key Error</td>
<td>KEIF</td>
<td>KEIE</td>
</tr>
</tbody>
</table>
41.5 OTPDEC application information

41.5.1 OTPDEC initialization process

Introduction

One key aspect of OTPDEC is the trusted initialization of its registers, as it involves secret keys and critical options like MODE bits.

Two trusted initialization schemes are recommended here below.

Note: Those sequence are for production code, as during firmware development it is not always recommended to lock the key or the region configuration.

 Writes to configuration registers are effective when the configuration locks allow it, even if the region is enabled.

One key for all regions initialization scheme

In this scheme one entity owns the secret key used to decrypt the four protected regions. The recommended OTPDEC configuration sequence is described below:

1. For \( x = 1 \) to 4 write the correct MODE[1:0] value in RxCFGR register.
2. For \( x = 1 \) to 4 program RxKEYR registers using the sequence described in KEYCRC (to have a valid CRC). Warning key registers are write only!
3. For \( x = 1 \) to 4 check the key CRC. If OK set KEYLOCK bit in RxCFGR register. This bit cannot be cleared, i.e. the key registers in this region \( x \) are no more writable.
4. Do while you have a region \( x \) to decrypt. This task does not necessarily have to be performed by the entity that owns the decryption keys.
   a) Verify if the key CRC corresponds to the encrypted binary stored in the region.
   b) Fill the detailed information corresponding to this binary (nonce, start address, end address, version number).
   c) Enable decryption of this region using REG_EN.
   d) Set CONFIGLOCK bit in RxCFGR. This bit cannot be cleared, i.e. the region configuration is no more writable.

Caution: For a given region, when MODE bits are changed the key registers and associated CRC are cleared by hardware. As a consequence step 1 above must be done before step 2, and MODE bits must not be modified after step 2.
One key per region initialization scheme

In this scheme one entity can own the secret used to decrypt one (or more) protected region. The recommended OTFDEC configuration sequence is described below:

Do while you have a region x to decrypt. This task must be performed by the entity that owns the corresponding key.

a) Write the correct MODE[1:0] value in RxCFGR register.

b) Program RxKEYR registers using the sequence described in KEYCRC (to have a valid key CRC). Warning key registers are write only!

c) Check the key CRC. If OK set KEYLOCK bit in RxCFGR register. This bit cannot be cleared, i.e. the key registers are no more writable.

d) Fill the detailed information corresponding to the protected firmware (nonce, start address, end address, version number).

e) Enable decryption of this region using REG_EN.

f) Set CONFIGLOCK bit in RxCFGR. This bit cannot be cleared, i.e. the region configuration is no more writable.

Caution: For a given region, when MODE bits are changed the key registers and associated CRC are cleared by hardware. As a consequence step a) above must be done before step b), and MODE bits must not be modified after step b).

41.5.2 OTFDEC and power management

Each time OTFDEC is reset the correct key loading sequence described in Section 41.5.1 must be performed (in this case KEYCRC equals to zero in OTFDEC_RxCFGR registers).

It is recommended for application software to verify this point each time OTFDEC is reset by hardware.

41.5.3 Encrypting for OTFDEC

When MODE equals 00, 01 and 10

OTFDEC uses a standard AES in counter (CTR) mode to encrypt binary stored in a region with this MODE value. CTR chaining is defined in NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.

When this mode is selected, any AES compatible hardware accelerator or library can be used to encrypt those protected libraries. Definition and endianness of the AES inputs and outputs are defined in Section 41.3.4: AES in counter mode decryption.

For more details, refer to AN5281 application note available on www.st.com.

When MODE equals to 11

OTFDEC uses a proprietary layer of protection on top of standard AES in counter mode when processing a code stored in a protected region with MODE[1:0] = 11.

Enhanced encryption mode can be used to increase the robustness against tampering.
When it is selected, two encryption scenarios are proposed:

a) Use any device in RDP0 and a debugger to encrypt the code using a RSS service. Refer to AN5281 application note for more details.

b) During a firmware install or update, use the OTFDEC in the device to encrypt the target code. For more details, refer to AN4992 application note (install) and UM2262 user manual (update) available on www.st.com.

41.5.4 OTFDEC Key CRC source code

Below is the CRC source code that can be used to compare with the result of the computation provided by OTFDEC in KEYCRC bitfield after loading the keys in OTFDEC_RxKEYR registers.

```c
uint8_t getCRC(uint32_t * keyin)
{
    const uint8_t CRC7_POLY = 0x7;
    const uint32_t key_strobe[4] = {0xAA55AA55, 0x3, 0x18, 0xC0};
    uint8_t  i, j, k, crc = 0x0;
    uint32_t  keyval;

    for (j = 0; j < 4; j++)
    {
        keyval = *keyin+j;
        if (j == 0)
        {
            keyval ^= key_strobe[0];
        }
        else
        {
            keyval ^= (key_strobe[j] << 24) | (crc << 16) | (key_strobe[j] << 8) | crc;
        }

        for (i = 0, crc = 0; i < 32; i++)
        {
            k = (((crc >> 7) ^ (keyval >> (31-i))&0xF)) & 1;
            crc <<= 1;
            if (k)
            {
                crc ^= CRC7_POLY;
            }
        }
        crc^=0x55;
    }
    return crc;
}
```
41.6 OTFDEC registers

41.6.1 OTFDEC region x configuration register (OTFDEC_RxCFGR)

Address offset: 0x20 + 0x30 * (x - 1) (x = 1 to 4)
Reset value: 0x0000 0000

Writes are ignored if CONFIGLOCK bit is set to 1.

<table>
<thead>
<tr>
<th>REGx_VERSION[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 31:16 REGx_VERSION[15:0]: region firmware version</th>
</tr>
</thead>
<tbody>
<tr>
<td>This 16-bit bitfield must be correctly initialized before the region corresponding REG_EN bit is set in the RxCFGR register.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:8 KEYCRC[7:0]: region key 8-bit CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>When KEYLOCK=0, KEYCRC bitfield is automatically computed by hardware while loading the key of this region in this exact sequence: KEYR0 then KEYR1 then KEYR2 then finally KEYR3 (all written once). A new computation starts as soon as a new valid sequence is initiated, and KEYCRC is read as zero until a valid sequence is completed. When KEYLOCK=1, KEYCRC remains unchanged until the next reset.</td>
</tr>
<tr>
<td>CRC computation is an 8-bit checksum using the standard CRC-8-CCITT algorithm $X^8 + X^2 + X + 1$ (according the convention). Source code is available in this manual.</td>
</tr>
<tr>
<td>This field is read only.</td>
</tr>
<tr>
<td>Note: CRC information is updated only after the last bit of the key has been written.</td>
</tr>
</tbody>
</table>

| Bits 7:6 Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bits 5:4 MODE[1:0]: operating mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bitfield selects the OTFDEC operating mode for this region:</td>
</tr>
<tr>
<td>00: Only instruction accesses are decrypted.</td>
</tr>
<tr>
<td>01: Only data accesses are decrypted.</td>
</tr>
<tr>
<td>10: All read accesses are decrypted (instruction or data).</td>
</tr>
<tr>
<td>11: Only instruction accesses are decrypted, and enhanced encryption mode is activated.</td>
</tr>
<tr>
<td>When MODE is not equal to 11 the standard AES encryption mode is activated.</td>
</tr>
<tr>
<td>When either of the MODE bits are changed the region’s key and associated CRC are zeroed.</td>
</tr>
</tbody>
</table>

| Bit 3 Reserved, must be kept at reset value. |
41.6.2 OTFDEC region x start address register (OTFDEC2_RxSTARTADDR)

Address offset: 0x24 + 0x30 * (x - 1) (x = 1 to 4)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>REGx_START_ADDR[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:0 REGx_START_ADDR[31:0]: Region AXI start address
This register must be written before the region corresponding REG_EN bit in the RxCFGR register is set.
Writing this register while the region CONFIGLOCK bit in the RxCFGR register is set will be discarded.

Note: When determining the region the first 12 bits (LSB) and the last 4 bits (MSB) are ignored.
When this register is accessed in read the 4 MSB bits and the 12 LSB bits return zero.

41.6.3 OTFDEC region x end address register (OTFDEC_RxENDADDR)

Address offset: 0x28 + 0x30 * (x - 1) (x = 1 to 4)

Reset value: 0x0000 0FFF
41.6.4 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0)
Address offset: 0x2C + 0x30 * (x - 1) (x = 1 to 4)
Reset value: 0x0000 0000

Bits 31:0 REGx_NONCE[31:0]: Region nonce, bits [31:0]
This register must be written before the region corresponding REG_EN bit in the RxCFGR register is set.
Writing this register while the region CONFIGLOCK bit in the RxCFGR register is set will be discarded.

41.6.5 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1)
Address offset: 0x30 + 0x30 * (x - 1) (x = 1 to 4)
Reset value: 0x0000 0000

Bits 31:0 REGx_NONCE[63:32]: Region nonce, bits [63:32]
Refer to the OTFDEC_RxNONCER0 register for description of the NONCE[63:0] bitfield.
41.6.6  **OTFDEC region x key register 0 (OTFDEC_RxKEYR0)**

Address offset: 0x34 + 0x30 * (x -1) (x = 1 to 4)
Reset value: 0x0000 0000

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

Bits 31:0 **REGx_KEY[31:0]**: Region key, bits [31:0]

This register must be written before the region corresponding REG_EN bit in the RxCFGR register is set.

Reading this register returns a zero value. Writing this register while the region CONFIGLOCK or KEYLOCK bit is set in the RxCFGR register will be discarded.

**Note:** When application successfully changes MODE bits in RxCFGR register RxKEYR registers and associated KEYCRC are erased.

41.6.7  **OTFDEC region x key register 1 (OTFDEC_RxKEYR1)**

Address offset: 0x38 + 0x30 * (x -1) (x = 1 to 4)
Reset value: 0x0000 0000

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

Bits 31:0 **REGx_KEY[63:32]**: Region key, bits [63:32]

Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

41.6.8  **OTFDEC region x key register 2 (OTFDEC_RxKEYR2)**

Address offset: 0x3C + 0x30 * (x -1) (x = 1 to 4)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>
### OTFDEC region x key register 3 (OTFDEC_RxKEYR3)

**Address offset:** 0x40 + 0x30 *(x -1) (x = 1 to 4)

**Reset value:** 0x0000 0000

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

### OTFDEC interrupt status register (OTFDEC_ISR)

**Address offset:** 0x300

**Reset value:** 0x0000 0000

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

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

**Bits 31:3** Reserved, must be kept at reset value.
Bit 2 **KEIF**: Key Error Interrupt Flag status

This bit is set by hardware and read only by application. Bit is set when a read access occurs on any encrypted region following the reset of the key registers by an abort event (tamper detection, unauthorized debugger connection, untrusted boot, RDP level regression).

Bit is cleared when application sets in OTFDEC_ICR the corresponding bit to “1”.

0: OTFDEC is operating properly.
1: Read access detected on an enabled encrypted region following an abort event. OTFDEC returns a zeroed value for the read, and an optional interrupt is generated if bit KEIE is set to “1” in OTFDEC_IER register.

After KEIF is set any subsequent read to any enabled encrypted region returns a zeroed value. This state remains until OTFDEC keys are initialized again.

Bit 1 **XONEIF**: Execute-only execute-Never Error Interrupt Flag status

This bit is set by hardware and read only by application. Bit is set when a read access and not an instruction fetch is detected on any encrypted region with MODE bits set to 00 or 11. It is also set when an instruction fetch and not a read access is detected on any encrypted region with MODE bits set to 01.

Bit is cleared when application sets in OTFDEC_ICR the corresponding bit to “1”.

0: No execute-only error status. No interrupt pending.
1: Read access detected on one region with MODE bits set to 00 or 11, or execute access detected on one region with MODE bits set to 01. OTFDEC returns a zeroed value for the illegal access, and an optional interrupt is generated if bit XONEIE is set to “1” in OTFDEC_IER register.

Bit 0 **SEIF**: Security Error Interrupt Flag status

This bit is set by hardware and read only by application. Bit is set when at least one security error has been detected (illegal access to keys, illegal write on locked configuration).

Bit is cleared when application sets in OTFDEC_ICR the corresponding bit to “1”.

0: No security error status. No interrupt pending.
1: Security error flag status, with interrupt pending. Actual interrupt generation is dependent on OTFDEC_IER corresponding bit SEIE.
41.6.11 OTFDEC interrupt clear register (OTFDEC_ICR)

Address offset: 0x304
Reset value: 0x0000 0000

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

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

Bit 2 KEIF: Key Error Interrupt Flag clear
This bit is written by application, and always reads as 0.
0: KEIF flag status is not affected
1: KEIF flag status is cleared in OTFDEC_ISR register

Note: Clearing KEIF does not solve the source of the problem (bad key registers). To be able to read or execute again any encrypted region, OTFDEC key registers must properly initialized, again.

Bit 1 XONEIF: Execute-only execute-Never Error Interrupt Flag clear
This bit is written by application, and always reads as 0.
0: XONEIF flag status is not affected
1: XONEIF flag status is cleared in OTFDEC_ISR register

Bit 0 SEIF: Security Error Interrupt Flag clear
This bit is written by application, and always reads as 0.
0: SEIF flag status is not affected
1: SEIF flag status is cleared in OTFDEC_ISR register

41.6.12 OTFDEC interrupt enable register (OTFDEC_IER)

Address offset: 0x308
Reset value: 0x0000 0000

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

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

Bit 2 KEIE: Key Error Interrupt enable
This bit is written by application, and always reads as 0.
0: KEIE flag is not enabled
1: KEIE flag is enabled in OTFDEC_ISR register

Bit 1 XONEIE: Execute-only execute-Never Error interrupt enable
This bit is written by application, and always reads as 0.
0: XONEIE flag is not enabled
1: XONEIE flag is enabled in OTFDEC_ISR register

Bit 0 SEIE: Security Error interrupt enable
This bit is written by application, and always reads as 0.
0: SEIE flag is not enabled
1: SEIE flag is enabled in OTFDEC_ISR register
Bits 31:3  Reserved, must be kept at reset value.

Bit 2  **KEIE**: Key Error Interrupt Enable
This bit is read and written by application. It controls the OTFDEC interrupt generation when KEIF flag status is set.
- 0: Interrupt generation on key error flag KEIF is disabled (masked)
- 1: Interrupt generation on key error flag KEIF is enabled (not masked)

Bit 1  **XONEIE**: Execute-only execute-Never Error Interrupt Enable
This bit is read and written by application. It controls the OTFDEC interrupt generation when XONEIF flag status is set.
- 0: Interrupt generation on execute-only error XONEIF is disabled (masked)
- 1: Interrupt generation on execute-only error XONEIF is enabled (not masked)

Bit 0  **SEIE**: Security Error Interrupt Enable
This bit is read and written by application. It controls the OTFDEC interrupt generation when SEIF flag status is set.
- 0: Interrupt generation on security error SEIF is disabled (masked)
- 1: Interrupt generation on security error SEIF is enabled (not masked)
41.6.13 OTFDEC register map

Table 301 gives the summary OTFDEC register map and reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00 - 0x1C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>OTFDEC_ R1CFGR</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>REG1_VERSION[15:0]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>KEYCRC[7:0]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>MODE[1:0]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>KEYLOCK.</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>CONFIGLOCK.</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>REG_EN</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td>0x24</td>
<td>OTFDEC_ R1STARTADDR</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>REG1_START_ADD[31:0]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td>0x28</td>
<td>OTFDEC_ R1ENDADDR</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>REG1_END_ADD[31:0]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td>0x2C</td>
<td>OTFDEC_ R1NONCER0</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>REG1_NONCE[31:0]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td>0x30</td>
<td>OTFDEC_ R1NONCER1</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>REG1_NONCE[83:32]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td>0x34</td>
<td>OTFDEC_ R1KEYR0</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>REG1_KEY[31:0]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td>0x38</td>
<td>OTFDEC_ R1KEYR1</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>REG1_KEY[53:32]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td>0x3C</td>
<td>OTFDEC_ R1KEYR2</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>REG1_KEY[95:64]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td>0x40</td>
<td>OTFDEC_REG1_ KEY3</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>REG1_KEY[127:96]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td>0x44 - 0x4C</td>
<td>Reserved</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td>0x50</td>
<td>OTFDEC_ R2CFGR</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>REG2_VERSION[15:0]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>KEYCRC[7:0]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>MODE[1:0]</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>KEYLOCK.</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>CONFIGLOCK.</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
<tr>
<td></td>
<td>REG_EN</td>
<td></td>
<td>0x00 - 0x1C</td>
</tr>
</tbody>
</table>
### Table 301. OTFDEC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x054</td>
<td>OTFDEC_R2STARTADDR</td>
<td>REG2_START_ADD[31:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x058</td>
<td>OTFDEC_R2ENADDR</td>
<td>REG2_END_ADD[31:0]</td>
<td>00000000000000000000111111111111</td>
</tr>
<tr>
<td>0x05C</td>
<td>OTFDEC_R2NONCER0</td>
<td>REG2_NONCE[31:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x060</td>
<td>OTFDEC_R2NONCER1</td>
<td>REG2_NONCE[63:32]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x064</td>
<td>OTFDEC_R2KEYR0</td>
<td>REG2_KEY[31:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x068</td>
<td>OTFDEC_R2KEYR1</td>
<td>REG2_KEY[63:32]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x06C</td>
<td>OTFDEC_R2KEYR2</td>
<td>REG2_KEY[95:64]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x070</td>
<td>OTFDEC_R2KEYR3</td>
<td>REG2_KEY[127:96]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x074 - 0x7C</td>
<td>Reserved</td>
<td>Reset value</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x80</td>
<td>OTFDEC_R3CFGR</td>
<td>REG3_VERSION[15:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x84</td>
<td>OTFDEC_R3STARTADDR</td>
<td>REG3_START_ADD[31:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x88</td>
<td>OTFDEC_R3ENADDR</td>
<td>REG3_END_ADD[31:0]</td>
<td>00000000000000000000111111111111</td>
</tr>
<tr>
<td>0x8C</td>
<td>OTFDEC_R3NONCER0</td>
<td>REG3_NONCE[31:0]</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x90</td>
<td>OTFDEC_R3NONCER1</td>
<td>REG3_NONCE[63:32]</td>
<td>00000000000000000000000000000000</td>
</tr>
</tbody>
</table>
Table 301. OTFDEC register map and reset values (continued)

| Offset | Register name        | Register name     | Offset | Reset value | Offset | Reset value | Offset | Reset value | Offset | Reset value | Offset | Reset value | Offset | Reset value | Offset | Reset value | Offset | Reset value | Offset | Reset value |
|--------|----------------------|-------------------|--------|-------------|--------|-------------|--------|-------------|--------|-------------|--------|-------------|--------|-------------|--------|-------------|--------|-------------|
| 0xB0   | OTFDEC_R4CFGFR       | REG4_CONFIG[15:0] | 0xB4   | REG4_CFGFR  | 0x9C   | REG3_KEY[95:64] | 0xA0   | REG3_KEY[127:96] | 0xA4   | Reserved | 0xA4   | Reserved | 0xA4   | Reserved | 0xA4   | Reserved |
Table 301. OTFDEC register map and reset values (continued)

| Offset  | Register name | reset value | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|---------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x0D4 - 0x2FC | Reserved | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x300 | OTFDEC_ISR | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x304 | OTFDEC_ICR | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x308 | OTFDEC_IER | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x30C - 0x3E8 | Reserved | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 on page 128 for the register boundary addresses.
42 Advanced-control timers (TIM1/TIM8)

42.1 TIM1/TIM8 introduction

The advanced-control timers (TIM1/TIM8) consist of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1/TIM8) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in Section 42.3.26: Timer synchronization.

42.2 TIM1/TIM8 main features

TIM1/TIM8 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also "on the fly") the counter clock frequency either by any factor between 1 and 65536.
- Up to 6 independent channels for:
  - Input Capture (but channels 5 and 6)
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management
1. See Figure 356: Break and Break2 circuitry overview for details.
42.3 TIM1/TIM8 functional description

42.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:
- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 314 and Figure 315 give some examples of the counter behavior when the prescaler ratio is changed on the fly:
Figure 314. Counter timing diagram with prescaler division change from 1 to 2

Figure 315. Counter timing diagram with prescaler division change from 1 to 4
42.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.
Figure 316. Counter timing diagram, internal clock divided by 1

Figure 317. Counter timing diagram, internal clock divided by 2
Figure 318. Counter timing diagram, internal clock divided by 4

Figure 319. Counter timing diagram, internal clock divided by N
Figure 320. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

Figure 321. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)
**Downcounting mode**

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn’t change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.
**Figure 322. Counter timing diagram, internal clock divided by 1**

CK_PSC

CNT_EN

Timerclock = CK_CNT

Counter register

Counter underflow

Update event (UEV)

Update interrupt flag (UIF)

**Figure 323. Counter timing diagram, internal clock divided by 2**

CK_PSC

CNT_EN

Timerclock = CK_CNT

Counter register

Counter underflow

Update event (UEV)

Update interrupt flag (UIF)
Figure 324. Counter timing diagram, internal clock divided by 4

Figure 325. Counter timing diagram, internal clock divided by N
Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to ‘00’. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or...
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 327. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6

1. Here, center-aligned mode 1 is used (for more details refer to **Section 42.4: TIM1/TIM8 registers**).
Figure 328. Counter timing diagram, internal clock divided by 2

Figure 329. Counter timing diagram, internal clock divided by 4, TIMx.ARR=0x36

Note: Here, center_aligned mode 2 or 3 is updated with an UIF on overflow
Figure 330. Counter timing diagram, internal clock divided by N

Figure 331. Counter timing diagram, update event with ARPE=1 (counter underflow)
42.3.3 Repetition counter

Section 42.3.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every \( N+1 \) counter overflows or underflows, where \( N \) is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode.

Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is \( 2 \times T_{ck} \), due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to Figure 333). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.
In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 333. Update rate examples depending on mode and TIMx_RCR register settings

**Counter-aligned mode**

- TIMx_RCR = 0
- UEV = Upcounting

**Edge-aligned mode**

- TIMx_RCR = 0
- UEV = Downcounting

- TIMx_RCR = 1
- UEV = Upcounting

- TIMx_RCR = 2
- UEV = Downcounting

- TIMx_RCR = 3
- UEV = Upcounting

- TIMx_RCR = 3 and re-synchronization
  - UEV = Updated event (by SW)

**UEV**: Update event: Preload registers transferred to active registers and update interrupt generated.

**Update Event**: If the repetition counter underflow occurs when the counter is equal to the auto-reload value.
42.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:
- external clock (external clock mode 2, see Section 42.3.5)
- trigger for the slave mode (see Section 42.3.26)
- PWM reset input for cycle-by-cycle current regulation (see Section 42.3.7)

Figure 334 below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield.

Figure 334. External trigger input block

The ETR input comes from multiple sources: input pins (default configuration), comparator outputs and analog watchdogs. The selection is done with the ETRSEL[3:0] bitfield.

Figure 335. TIM1/TIM8 ETR input circuitry
42.3.5 **Clock selection**

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

**Internal clock source (CK_INT)**

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

*Figure 336* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

*Figure 336. Control circuit in normal mode, internal clock divided by 1*

![Control Circuit Diagram]

**External clock source mode 1**

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.
1. Codes ranging from 01000 to 11111 are reserved

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

**Note:** The capture prescaler is not used for triggering, so the user does not need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The Figure 339 gives an overview of the external trigger input block.

1. Refer to Figure 335: TIM1/TIM8 ETR input circuitry.

For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:
1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by proper ETPS prescaler setting.

Figure 340. Control circuit in external clock mode 2

![Control circuit diagram](image-url)
42.3.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

*Figure 341 to Figure 344* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

*Figure 341. Capture/compare channel (example: channel 1 input stage)*

The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

*Figure 342. Capture/compare channel 1 main circuit*
Figure 343. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)

1. OCxREF, where x is the rank of the complementary channel

Figure 344. Output stage of capture/compare channel (channel 4)
1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 42.3.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when written with '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been
detected (sampled at \(f_{\text{DTS}}\) frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

4. Select the edge of the active transition on the TI1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).

5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ’00’ in the TIMx_CCMR1 register).

6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.

7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:
- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

*Note:* IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

### 42.3.8 PWM input mode

This mode is a particular case of input capture mode. The procedure is the same except:
- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, the user can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):
1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP='10' (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 346. PWM input mode timing

42.3.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCxREF/OCx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCxREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.
42.3.10 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while Channel 5 and 6 are only available inside the device (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM=0000), be set active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=’0’, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 347.
### 42.3.11 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '0110' (PWM mode 1) or '0111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxE bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.
PWM edge-aligned mode

- **Upcounting configuration**
  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the *Upcounting mode on page 1443*.
  
  In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. Figure 348 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

- **Downcounting configuration**
  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Downcounting mode on page 1447*.
  
  In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF is held at ‘1’. 0% PWM is not possible in this mode.

PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the
TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to the Center-aligned mode (up/down counting) on page 1450.

Figure 349 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Hints on using center-aligned mode

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit.
in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.

- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

### 42.3.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing '1110' (Asymmetric PWM mode 1) or '1111' (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

**Note:** The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 1.

*Figure 350* represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.
42.3.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 351 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1.
42.3.14 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The OC5REF signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the OC5REF is combined. The resulting signals, OCxREFC, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, OC1REFC is controlled by TIMx_CCR1 and TIMx_CCR5
- If GC5C2 is set, OC2REFC is controlled by TIMx_CCR2 and TIMx_CCR5
- If GC5C3 is set, OC3REFC is controlled by TIMx_CCR3 and TIMx_CCR5

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].
The TRGO2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to Section 42.3.27: ADC synchronization for more details.

### 42.3.15 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to Table 306: Output control bits for complementary OCx and OCxN channels with break feature on page 1519 for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

**Figure 353. Complementary output with dead-time insertion**

**Figure 354. Dead-time waveforms with delay greater than the negative pulse**
The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to Section 42.4.20: TIMx break and dead-time register (TIMx_BDTR)\( (x = 1, 8) \) for delay calculation.

### Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

**Note:** When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

### 42.3.16 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM1 and TIM8 timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.
The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows the outputs to be enabled/disabled by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to Table 306: Output control bits for complementary OCx and OCxN channels with break feature on page 1519 for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKE/BK2E and BKP/BK2P can be modified at the same time. When the BKE/BK2E and BKP/BK2P bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The sources for break (BRK) channel are:
- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source:
  - the output from a comparator, with polarity selection and optional digital filtering
  - the analog watchdog output of the DFSDM1 peripheral
- A system break:
  - the Cortex®-M7 LOCKUP output
  - the PVD output
  - all SRAM and TCM double ECC errors (AXI-SRAM, ITCM, DTCM, SRAM1, SRAM2, SRAM3, SRAM4, BKRAM, refer to SYSCFG_CFGR register for details)
  - a Flash double ECC error
  - a clock failure event generated by the CSS detector

The sources for break2 (BRK2) are:
- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source coming from a comparator output.
- The analog watchdog output of the DFSDM1 peripheral

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register. The software break generation using BG and B2G is active whatever the BKE and BK2E enable bits values.
All sources are ORed before entering the timer BRK or BRK2 inputs, as per Figure 356 below.

**Figure 356. Break and Break2 circuitry overview**

*Note:* An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.
When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSS1 bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSS1=0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 ck_tim clock cycles).
  - If OSS1=0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (SBIF, BIF and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to ‘1’ again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be frozen (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to Section 42.4.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8). The LOCK bits can be written only once after an MCU reset.

Figure 357 shows an example of behavior of the outputs in response to a break.
Figure 357. Various output behavior in response to a break event on BRK (OSSI = 1)

- OCxREF
- OCx (OCxN not implemented, CCxP=0, OISx=1)
- OCx (OCxN not implemented, CCxP=0, OISx=0)
- OCx (OCxN not implemented, CCxP=1, OISx=1)
- OCx (OCxN not implemented, CCxP=1, OISx=0)
- OCx
- OCxN (CCxE=1, CCxP=0, OISx=0, CCxNE=1, CCxNP=0, OISxN=1)
- OCxN (CCxE=1, CCxP=0, OISx=1, CCxNE=1, CCxNP=1, OISxN=1)
- OCxN (CCxE=1, CCxP=0, OISx=0, CCxNE=0, CCxNP=0, OISxN=1)
- OCxN (CCxE=1, CCxP=0, OISx=1, CCxNE=0, CCxNP=0, OISxN=0)
- OCxN (CCxE=1, CCxP=0, CCxNE=0, CCxNP=0, OISx=OISxN=0 or OISx=OISxN=1)
The two break inputs have different behaviors on timer outputs:
- The BRK input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- BRK2 can only disable (inactive state) the PWM outputs.

The BRK has a higher priority than BRK2 input, as described in Table 302.

*Note:* BRK2 must only be used with OSSR = OSSI = 1.

**Table 302. Behavior of timer outputs versus BRK/BRK2 inputs**

<table>
<thead>
<tr>
<th>BRK</th>
<th>BRK2</th>
<th>Timer outputs state</th>
<th>Typical use case</th>
<th>OCxN output (low side switches)</th>
<th>OCx output (high side switches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>X</td>
<td>– Inactive then forced output state (after a deadtime)</td>
<td>ON after deadtime insertion</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Outputs disabled if OSSI = 0 (control taken over by GPIO logic)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Inactive Active Inactive OFF OFF

**Figure 358** gives an example of OCx and OCxN output behavior in case of active signals on BRK and BRK2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

**Figure 358. PWM output state following BRK and BRK2 pins assertion (OSSI=1)**
42.3.17 Bidirectional break inputs

The TIM1/TIM8 are featuring bidirectional break I/Os, as represented on Figure 360.

They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break and break2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the break and break2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to ‘0’ to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BK(2)E = 1). When a software break event is generated with BK(2)E = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break(2) I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which forces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BK(2)DSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see Table 303)
Arming and re-arming break circuitry

The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Table 303. Break protection disarming conditions

<table>
<thead>
<tr>
<th>MOE</th>
<th>BKDIR (BK2DIR)</th>
<th>BKDSRM (BK2DSRM)</th>
<th>Break protection state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Disarmed</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Armed</td>
</tr>
</tbody>
</table>

42.3.18 Clearing the OCxREF signal on an external event

The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be
used in Output compare and PWM modes. It does not work in Forced mode. The ocref_clr_int is connected to the ETRF signal (ETR after filtering).

When ETRF is chosen, ETR must be configured as follows:
1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to '00'.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to '0'.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

*Figure 361* shows the behavior of the OCxREF signal when the ETRF Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

*Figure 361. Clearing TIMx OCxREF*

---

**Note:** In case of a PWM with a 100% duty cycle (if CCRx>ARR), then OCxREF is enabled again at the next counter overflow.
### 42.3.19 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 362* describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

*Figure 362. 6-step generation, COM example (OSSR=1)*
### 42.3.20 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

For example one may want to generate a positive pulse on OC1 with a length of \( t_{\text{PULSE}} \) and after a delay of \( t_{\text{DELAY}} \) as soon as a positive edge is detected on the TI2 input pin.

Let’s use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to ‘110’ in the TIMx_SMCR register (trigger mode).
The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The \( t_{\text{DELAY}} \) is defined by the value written in the TIMx_CCR1 register.
- The \( t_{\text{PULSE}} \) is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on Ti1 input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay \( t_{\text{DELAY \, min}} \) we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 42.3.21 Retriggerable one pulse mode (OPM)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in Section 42.3.20:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

**Note:** The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.
42.3.22 Encoder interface mode

To select Encoder Interface mode write SMS='001' in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS='010' if it is counting on TI1 edges only and SMS='011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to a quadrature encoder. Refer to Table 304. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to ‘1’). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

Note: The prescaler must be set to zero when encoder mode is enabled

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder’s position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.
A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally used to convert the encoder’s differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The Figure 365 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S=’01’ (TIMx_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S=’01’ (TIMx_CCMR2 register, TI1FP2 mapped on TI2).
- CC1P=’0’ and CC1NP=’0’ (TIMx_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P=’0’ and CC2NP=’0’ (TIMx_CCER register, TI1FP2 non-inverted, TI1FP2= TI2).
- SMS=’011’ (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN=’1’ (TIMx_CR1 register, Counter enabled).

**Table 304. Counting direction versus encoder signals**

<table>
<thead>
<tr>
<th>Active edge</th>
<th>Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)</th>
<th>TI1FP1 signal</th>
<th>TI2FP2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rising</td>
<td>Falling</td>
<td>Rising</td>
</tr>
<tr>
<td>Counting on TI1 only</td>
<td>High</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>Down</td>
</tr>
<tr>
<td>Counting on TI2 only</td>
<td>High</td>
<td>No Count</td>
<td>No Count</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>No Count</td>
<td>No Count</td>
</tr>
<tr>
<td>Counting on TI1 and TI2</td>
<td>High</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>Down</td>
</tr>
</tbody>
</table>

![Figure 365. Example of counter operation in encoder interface mode.](MS33107V1)
Figure 366 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

![Figure 366. Example of encoder interface mode with TI1FP1 polarity inverted.](image)

The timer, when configured in Encoder Interface mode provides information on the sensor’s current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register’s bit 31 (TIMx_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter’s most significant bit is only accessible in write mode).

### 42.3.23 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register’s bit 31 (TIMx_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the UIF and UIFCPY flags assertion.
42.3.24 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per Figure 367 below.

![Figure 367. Measuring time interval between edges on 3 signals](image)

42.3.25 Interfacing with Hall sensors

This is done using the advanced-control timers (TIM1 or TIM8) to generate PWM signals to drive the motor and another timer TIMx (TIM2, TIM3, TIM4) referred to as “interfacing timer” in Figure 368. The “interfacing timer” captures the 3 timer input pins (CC1, CC2, CC3) connected through a XOR to the TI1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is TI1F_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TRC (See Figure 341: Capture/compare channel (example: channel 1 input stage) on page 1461). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1 or TIM8) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1 or TIM8) through the TRGO output.
Example: one wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’,
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx_CCMR1 register to ‘01’ The digital filter can also be programmed if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the TIMx_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx_CR2 register to ‘101’,

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

The Figure 368 describes this example.
Figure 368. Example of Hall sensor interface
42.3.26 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. They can be synchronized in several modes: Reset mode, Gated mode, and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 369. Control circuit in reset mode
**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).

- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

![Figure 370. Control circuit in Gated mode](image)

**Slave mode: Trigger mode**

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1
register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

**Figure 371. Control circuit in trigger mode**

![Control circuit in trigger mode](image)

**Slave mode: Combined reset + trigger mode**

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

**Slave mode: external clock mode 2 + trigger mode**

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:
1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.

2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01 in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP='0' in TIMx_CCRER register to validate the polarity (and detect rising edge only).

3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

**Figure 372. Control circuit in external clock mode 2 + trigger mode**

**Note:** The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.
42.3.27 ADC synchronization

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the TRGO2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in Figure 352 on page 1473.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

42.3.28 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.
This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.

2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:
   DBL = 3 transfers, DBA = 0xE.

3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).

4. Enable TIMx

5. Enable the DMA channel

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

42.3.29 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).
42.4 TIM1/TIM8 registers

Refer to for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

42.4.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8)

Address offset: 0x00

Reset value: 0x0000

| Bit 15:12 | Reserved, must be kept at reset value. |
| Bit 11 | UIFREMAP: UIF status bit remapping |
| 0 | No remapping. UIF status bit is not copied to TIMx_CNT register bit 31. |
| 1 | Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31. |
| Bit 10 | Reserved, must be kept at reset value. |
| Bits 9:8 | CKD[1:0]: Clock division |
| 00 | \( t_{DTS}=t_{CK\_INT} \) |
| 01 | \( t_{DTS}=2t_{CK\_INT} \) |
| 10 | \( t_{DTS}=4t_{CK\_INT} \) |
| 11 | Reserved, do not program this value |
| Bit 7 | ARPE: Auto-reload preload enable |
| 0 | TIMx_ARR register is not buffered |
| 1 | TIMx_ARR register is buffered |
| Bits 6:5 | CMS[1:0]: Center-aligned mode selection |
| 00 | Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR). |
| 01 | Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down. |
| 10 | Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up. |
| 11 | Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down. |

Note: Switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1) is not allowed.
42.4.2 TIMx control register 2 (TIMx.CR2)(x = 1, 8)

Address offset: 0x04

Reset value: 0x0000 0000
Bits 31:24  Reserved, must be kept at reset value.

Bits 23:20  MMS2[3:0]: Master mode selection 2
These bits allow the information to be sent to ADC for synchronization (TRGO2) to be selected. The combination is as follows:
0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on TRGO2 is delayed compared to the actual reset.
0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).
0010: **Update** - the update event is selected as trigger output (TRGO2). For instance, a master timer can then be used as a prescaler for a slave timer.
0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (TRGO2).
0100: **Compare** - OC1REF signal is used as trigger output (TRGO2)
0101: **Compare** - OC2REF signal is used as trigger output (TRGO2)
0110: **Compare** - OC3REF signal is used as trigger output (TRGO2)
0111: **Compare** - OC4REF signal is used as trigger output (TRGO2)
1000: **Compare** - OC5REF signal is used as trigger output (TRGO2)
1001: **Compare** - OC6REF signal is used as trigger output (TRGO2)
1010: **Compare Pulse** - OC4REF rising or falling edges generate pulses on TRGO2
1011: **Compare Pulse** - OC6REF rising or falling edges generate pulses on TRGO2
1100: **Compare Pulse** - OC4REF or OC6REF rising edges generate pulses on TRGO2
1101: **Compare Pulse** - OC4REF rising or OC6REF falling edges generate pulses on TRGO2
1110: **Compare Pulse** - OC5REF or OC6REF rising edges generate pulses on TRGO2
1111: **Compare Pulse** - OC5REF rising or OC6REF falling edges generate pulses on TRGO2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19  Reserved, must be kept at reset value.

Bit 18  OIS6: Output Idle state 6 (OC6 output)
Refer to OIS1 bit

Bit 17  Reserved, must be kept at reset value.

Bit 16  OIS5: Output Idle state 5 (OC5 output)
Refer to OIS1 bit

Bit 15  Reserved, must be kept at reset value.

Bit 14  OIS4: Output Idle state 4 (OC4 output)
Refer to OIS1 bit

Bit 13  OIS3N: Output Idle state 3 (OC3N output)
Refer to OIS1N bit

Bit 12  OIS3: Output Idle state 3 (OC3 output)
Refer to OIS1 bit
Bit 11 **OIS2N**: Output Idle state 2 (OC2N output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output Idle state 2 (OC2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)
0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0

*Note:* This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)
0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

*Note:* This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: TI1 selection
0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits allow selected information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred.
100: **Compare** - OC1REF signal is used as trigger output (TRGO)
101: **Compare** - OC2REF signal is used as trigger output (TRGO)
110: **Compare** - OC3REF signal is used as trigger output (TRGO)
111: **Compare** - OC4REF signal is used as trigger output (TRGO)

*Note:* The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs
Bit 2  **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI

*Note*: This bit acts only on channels that have a complementary output.

Bit 1  Reserved, must be kept at reset value.

Bit 0  **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).

*Note*: This bit acts only on channels that have a complementary output.

## 42.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)

**Address offset**: 0x08

**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:22</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 19:17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

**Bit 15  **ETP**: External trigger polarity
This bit selects whether ETR or ETR̅ is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge.
1: ETR is inverted, active at low level or falling edge.

**Bit 14  **ECE**: External clock enable
This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

*Note*: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=0011).

It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 0011).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.
Bits 13:12  **ETPS[1:0]:** External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Prescaler OFF</td>
</tr>
<tr>
<td>01</td>
<td>ETRP frequency divided by 2</td>
</tr>
<tr>
<td>10</td>
<td>ETRP frequency divided by 4</td>
</tr>
<tr>
<td>11</td>
<td>ETRP frequency divided by 8</td>
</tr>
</tbody>
</table>

Bits 11:8  **ETF[3:0]:** External trigger filter

This bit-field defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>No filter, sampling is done at fDTS</td>
</tr>
<tr>
<td>0001</td>
<td>fSAMPLING=fCK_INT, N=2</td>
</tr>
<tr>
<td>0010</td>
<td>fSAMPLING=fCK_INT, N=4</td>
</tr>
<tr>
<td>0011</td>
<td>fSAMPLING=fCK_INT, N=8</td>
</tr>
<tr>
<td>0100</td>
<td>fSAMPLING=fDTS/2, N=6</td>
</tr>
<tr>
<td>0101</td>
<td>fSAMPLING=fDTS/2, N=8</td>
</tr>
<tr>
<td>0110</td>
<td>fSAMPLING=fDTS/4, N=6</td>
</tr>
<tr>
<td>0111</td>
<td>fSAMPLING=fDTS/4, N=8</td>
</tr>
<tr>
<td>1000</td>
<td>fSAMPLING=fDTS/8, N=6</td>
</tr>
<tr>
<td>1001</td>
<td>fSAMPLING=fDTS/8, N=8</td>
</tr>
<tr>
<td>1010</td>
<td>fSAMPLING=fDTS/16, N=5</td>
</tr>
<tr>
<td>1011</td>
<td>fSAMPLING=fDTS/16, N=6</td>
</tr>
<tr>
<td>1100</td>
<td>fSAMPLING=fDTS/16, N=8</td>
</tr>
<tr>
<td>1101</td>
<td>fSAMPLING=fDTS/32, N=5</td>
</tr>
<tr>
<td>1110</td>
<td>fSAMPLING=fDTS/32, N=6</td>
</tr>
<tr>
<td>1111</td>
<td>fSAMPLING=fDTS/32, N=8</td>
</tr>
</tbody>
</table>

Bit 7  **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4  **TS[4:0]:** Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>Internal Trigger 0 (ITR0)</td>
</tr>
<tr>
<td>00001</td>
<td>Internal Trigger 1 (ITR1)</td>
</tr>
<tr>
<td>00010</td>
<td>Internal Trigger 2 (ITR2)</td>
</tr>
<tr>
<td>00011</td>
<td>Internal Trigger 3 (ITR3)</td>
</tr>
<tr>
<td>00100</td>
<td>TI1 Edge Detector (TI1F_ED)</td>
</tr>
<tr>
<td>00101</td>
<td>Filtered Timer Input 1 (TI1FP1)</td>
</tr>
<tr>
<td>00110</td>
<td>Filtered Timer Input 2 (TI2FP2)</td>
</tr>
<tr>
<td>00111</td>
<td>External Trigger input (ETRF)</td>
</tr>
</tbody>
</table>

Others: Reserved

See Table 305: TIMx internal trigger connection on page 1505 for more details on ITRx meaning for each Timer.

**Note:** These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3  Reserved, must be kept at reset value.
Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.
0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Codes above 1000: Reserved.

**Note:** The gated mode must not be used if Ti1F_ED is selected as the trigger input (TS=00100). Indeed, Ti1F_ED outputs 1 pulse for each transition on Ti1F, whereas the gated mode checks the level of the trigger signal.

**Note:** The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

### Table 305. TIMx internal trigger connection

<table>
<thead>
<tr>
<th>Slave TIM</th>
<th>ITR0 (TS = 00000)</th>
<th>ITR1 (TS = 00001)</th>
<th>ITR2 (TS = 00010)</th>
<th>ITR3 (TS = 00011)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM1</td>
<td>TIM15</td>
<td>TIM2</td>
<td>TIM3</td>
<td>TIM4</td>
</tr>
<tr>
<td>TIM8</td>
<td>TIM1</td>
<td>TIM2</td>
<td>TIM4</td>
<td>TIM5</td>
</tr>
</tbody>
</table>

### 42.4.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8)

Address offset: 0x0C
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDE</td>
<td>COMDE</td>
<td>CC4DE</td>
<td>CC3DE</td>
<td>CC2DE</td>
<td>CC1DE</td>
<td>UDE</td>
<td>BIE</td>
<td>TIE</td>
<td>COMIE</td>
<td>CC4IE</td>
<td>CC3IE</td>
<td>CC2IE</td>
<td>CC1IE</td>
<td>UIE</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>
Bit 15  Reserved, must be kept at reset value.

Bit 14  **TDE**: Trigger DMA request enable
         0: Trigger DMA request disabled
         1: Trigger DMA request enabled

Bit 13  **COMDE**: COM DMA request enable
         0: COM DMA request disabled
         1: COM DMA request enabled

Bit 12  **CC4DE**: Capture/Compare 4 DMA request enable
         0: CC4 DMA request disabled
         1: CC4 DMA request enabled

Bit 11  **CC3DE**: Capture/Compare 3 DMA request enable
         0: CC3 DMA request disabled
         1: CC3 DMA request enabled

Bit 10  **CC2DE**: Capture/Compare 2 DMA request enable
         0: CC2 DMA request disabled
         1: CC2 DMA request enabled

Bit 9   **CC1DE**: Capture/Compare 1 DMA request enable
         0: CC1 DMA request disabled
         1: CC1 DMA request enabled

Bit 8   **UDE**: Update DMA request enable
         0: Update DMA request disabled
         1: Update DMA request enabled

Bit 7   **BIE**: Break interrupt enable
         0: Break interrupt disabled
         1: Break interrupt enabled

Bit 6   **TIE**: Trigger interrupt enable
         0: Trigger interrupt disabled
         1: Trigger interrupt enabled

Bit 5   **COMIE**: COM interrupt enable
         0: COM interrupt disabled
         1: COM interrupt enabled

Bit 4   **CC4IE**: Capture/Compare 4 interrupt enable
         0: CC4 interrupt disabled
         1: CC4 interrupt enabled

Bit 3   **CC3IE**: Capture/Compare 3 interrupt enable
         0: CC3 interrupt disabled
         1: CC3 interrupt enabled
Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

### 42.4.5 TIMx status register (TIMx_SR)(x = 1, 8)

Address offset: 0x10

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-18</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>CC6IF</td>
<td>Compare 6 interrupt flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to CC1IF description (Note: Channel 6 can only be configured as output)</td>
</tr>
<tr>
<td>16</td>
<td>CC5IF</td>
<td>Compare 5 interrupt flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to CC1IF description (Note: Channel 5 can only be configured as output)</td>
</tr>
<tr>
<td>15-14</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>13</td>
<td>SBIF</td>
<td>System Break interrupt flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This flag is set by hardware as soon as the system break input goes active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It can be cleared by software if the system break input is not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This flag must be reset to re-start PWM operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: No break event occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.</td>
</tr>
<tr>
<td>12</td>
<td>CC4OF</td>
<td>Capture/Compare 4 overcapture flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to CC1OF description</td>
</tr>
<tr>
<td>11</td>
<td>CC3OF</td>
<td>Capture/Compare 3 overcapture flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to CC1OF description</td>
</tr>
<tr>
<td>10</td>
<td>CC2OF</td>
<td>Capture/Compare 2 overcapture flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to CC1OF description</td>
</tr>
<tr>
<td>Bit 9</td>
<td>CC10F: Capture/Compare 1 overcapture flag</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No overcapture has been detected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 8</th>
<th>B2IF: Break 2 interrupt flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.</td>
</tr>
<tr>
<td></td>
<td>0: No break event occurred.</td>
</tr>
<tr>
<td></td>
<td>1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx_DIER register.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>BIF: Break interrupt flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.</td>
</tr>
<tr>
<td></td>
<td>0: No break event occurred.</td>
</tr>
<tr>
<td></td>
<td>1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>TIF: Trigger interrupt flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: No trigger event occurred.</td>
</tr>
<tr>
<td></td>
<td>1: Trigger interrupt pending.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 5</th>
<th>COMIF: COM interrupt flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: No COM event occurred.</td>
</tr>
<tr>
<td></td>
<td>1: COM interrupt pending.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>CC4IF: Capture/Compare 4 interrupt flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Refer to CC1IF description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>CC3IF: Capture/Compare 3 interrupt flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Refer to CC1IF description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>CC2IF: Capture/Compare 2 interrupt flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Refer to CC1IF description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>CC1IF: Capture/Compare 1 interrupt flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).</td>
</tr>
<tr>
<td></td>
<td>0: No compare match / No input capture occurred</td>
</tr>
<tr>
<td></td>
<td>1: A compare match or an input capture occurred.</td>
</tr>
</tbody>
</table>

If channel CC1 is configured as output: this flag is set when he content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).
Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
- At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by a trigger event (refer to Section 42.4.3: TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)), if URS=0 and UDIS=0 in the TIMx_CR1 register.

42.4.6 **TIMx event generation register (TIMx_EGR)(x = 1, 8)**

Address offset: 0x14
Reset value: 0x0000

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B2G</td>
<td>BG</td>
<td>TG</td>
<td>COMG</td>
<td>CC4G</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

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

Bit 8 **B2G**: Break 2 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When CCPC bit is set, it allows CCxE, CCxNE and OCxM bits to be updated.

*Note: This bit acts only on channels having a complementary output.*

Bit 4 **CC4G**: Capture/Compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation
Refer to CC1G description
42.4.7 TIMx capture/compare mode register 1 [alternate]
(TIMx_CCMR1)(x = 1, 8)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Input capture mode:

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 IC2F[3:0]: Input capture 2 filter
Refer to IC1F[3:0] description.

Bits 11:10 IC2PSC[1:0]: Input capture 2 prescaler
Refer to OC1PSC[1:0] description.
Bits 9:8  **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).*

Bits 7:4  **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at fDTS
0001: fSAMPLING=fCK_INT, N=2
0010: fSAMPLING=fCK_INT, N=4
0011: fSAMPLING=fCK_INT, N=8
0100: fSAMPLING=fDTS/2, N=6
0101: fSAMPLING=fDTS/2, N=8
0110: fSAMPLING=fDTS/4, N=6
0111: fSAMPLING=fDTS/4, N=8
1000: fSAMPLING=fDTS/8, N=6
1001: fSAMPLING=fDTS/8, N=8
1010: fSAMPLING=fDTS/16, N=5
1011: fSAMPLING=fDTS/16, N=6
1100: fSAMPLING=fDTS/32, N=5
1101: fSAMPLING=fDTS/32, N=6
1110: fSAMPLING=fDTS/32, N=8
1111: fSAMPLING=fDTS/32, N=8

Bits 3:2  **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0  **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).*

**42.4.8 TIMx capture/compare mode register 1 [alternate]**

(TIMx_CCMR1)(x = 1, 8)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the
corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

**Output compare mode:**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>OC2</td>
<td>OC2M[2:0]</td>
<td>OC2 PE</td>
<td>OC2 FE</td>
<td>CC2S[1:0]</td>
<td>OC1 CE</td>
<td>OC1M[2:0]</td>
<td>OC1 PE</td>
<td>OC1 FE</td>
<td>CC1S[1:0]</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25  Reserved, must be kept at reset value.

Bits 23:17  Reserved, must be kept at reset value.

Bit 15  **OC2CE:** Output Compare 2 clear enable
Refer to OC1CE description.

Bits 24, 14:12  **OC2M[3:0]:** Output Compare 2 mode
Refer to OC1M[3:0] description.

Bit 11  **OC2PE:** Output Compare 2 preload enable
Refer to OC1PE description.

Bit 10  **OC2FE:** Output Compare 2 fast enable
Refer to OC1FE description.

Bits 9:8  **CC2S[1:0]:** Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).*

Bit 7  **OC1CE:** Output Compare 1 clear enable
0: OC1Ref is not affected by the ETRF input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input
Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. (this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0') as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF='1').

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

**Note:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

**Note:** In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

**Note:** On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.
Advanced-control timers (TIM1/TIM8) RM0455

42.4.9 TIMx capture/compare mode register 2 [alternate]
(TIMx_CCMR2)(x = 1, 8)

Address offset: 0x1C
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

**Input capture mode:**

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

**Output capture mode:**

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output). The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

**Bit 3 OC1PE:** Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

**Note:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

**Bit 2 OC1FE:** Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

**Bits 1:0 CC1S[1:0]: Capture/Compare 1 selection**

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

**Note:** CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).
Bits 31:16  Reserved, must be kept at reset value.

Bits 15:12  IC4F[3:0]: Input capture 4 filter
            Refer to IC1F[3:0] description.

Bits 11:10  IC4PSC[1:0]: Input capture 4 prescaler
            Refer to IC1PSC[1:0] description.

Bits 9:8  CC4S[1:0]: Capture/Compare 4 selection
            This bit-field defines the direction of the channel (input/output) as well as the used input.
            00: CC4 channel is configured as output
            01: CC4 channel is configured as input, IC4 is mapped on TI4
            10: CC4 channel is configured as input, IC4 is mapped on TI3
            11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
            Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bits 7:4  IC3F[3:0]: Input capture 3 filter
            Refer to IC1F[3:0] description.

Bits 3:2  IC3PSC[1:0]: Input capture 3 prescaler
            Refer to IC1PSC[1:0] description.

Bits 1:0  CC3S[1:0]: Capture/compare 3 selection
            This bit-field defines the direction of the channel (input/output) as well as the used input.
            00: CC3 channel is configured as output
            01: CC3 channel is configured as input, IC3 is mapped on TI3
            10: CC3 channel is configured as input, IC3 is mapped on TI4
            11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
            Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

42.4.10  TIMx capture/compare mode register 2 [alternate]
        (TIMx_CCMR2)(x = 1, 8)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

**Output compare mode**

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OC4CE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:25  Reserved, must be kept at reset value.

Bits 23:17  Reserved, must be kept at reset value.

Bit 15  **OC4CE**: Output compare 4 clear enable
Refer to OC1CE description.

Bits 24, 14:12  **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0] description.

Bit 11  **OC4PE**: Output compare 4 preload enable
Refer to OC1PE description.

Bit 10  **OC4FE**: Output compare 4 fast enable
Refer to OC1FE description.

Bits 9:8  **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).*

Bit 7  **OC3CE**: Output compare 3 clear enable
Refer to OC1CE description.

Bits 16, 6:4  **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M[3:0] description.

Bit 3  **OC3PE**: Output compare 3 preload enable
Refer to OC1PE description.

Bit 2  **OC3FE**: Output compare 3 fast enable
Refer to OC1FE description.

Bits 1:0  **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).*
42.4.11 TIMx capture/compare enable register
(TIMx_CCER)(x = 1, 8)

Address offset: 0x20

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CC4NP</td>
<td>CC4P</td>
<td>CC4E</td>
<td>CC3NP</td>
<td>CC3NE</td>
<td>CC3P</td>
<td>CC3E</td>
<td>CC2NP</td>
<td>CC2NE</td>
<td>CC2P</td>
<td>CC2E</td>
<td>CC1NP</td>
<td>CC1NE</td>
<td>CC1P</td>
<td>CC1E</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

- Bit 21 **CC6P**: Capture/Compare 6 output polarity
  - Refer to CC1P description
- Bit 20 **CC6E**: Capture/Compare 6 output enable
  - Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

- Bit 17 **CC5P**: Capture/Compare 5 output polarity
  - Refer to CC1P description
- Bit 16 **CC5E**: Capture/Compare 5 output enable
  - Refer to CC1E description
- Bit 15 **CC4NP**: Capture/Compare 4 complementary output polarity
  - Refer to CC1NP description
- Bit 14 Reserved, must be kept at reset value.
- Bit 13 **CC4P**: Capture/Compare 4 output polarity
  - Refer to CC1P description
- Bit 12 **CC4E**: Capture/Compare 4 output enable
  - Refer to CC1E description
- Bit 11 **CC3NP**: Capture/Compare 3 complementary output polarity
  - Refer to CC1NP description
- Bit 10 **CC3NE**: Capture/Compare 3 complementary output enable
  - Refer to CC1NE description
- Bit 9 **CC3P**: Capture/Compare 3 output polarity
  - Refer to CC1P description
- Bit 8 **CC3E**: Capture/Compare 3 output enable
  - Refer to CC1E description
- Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
  - Refer to CC1NP description
- Bit 6 **CC2NE**: Capture/Compare 2 complementary output enable
  - Refer to CC1NE description
Bit 5 **CC2P**: Capture/Compare 2 output polarity  
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable  
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity  
**CC1 channel configured as output:**  
0: OC1N active high.  
1: OC1N active low.  
**CC1 channel configured as input:**  
This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (channel configured as output).*

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable  
0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.  
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

*On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 1 **CC1P**: Capture/Compare 1 output polarity  
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)  
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)  
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

**CC1NP=0, CC1P=0**: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

**CC1NP=0, CC1P=1**: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

**CC1NP=1, CC1P=1**: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

**CC1NP=1, CC1P=0**: The configuration is reserved, it must not be used.  
*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.
Bit 0  **CC1E**: Capture/Compare 1 output enable  
0: Capture mode disabled / OC1 is not active (see below)  
1: Capture mode enabled / OC1 signal is output on the corresponding output pin  

When **CC1 channel is configured as output**, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to [Table 306](#) for details.  

**Note**: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

<table>
<thead>
<tr>
<th>Control bits</th>
<th>Output states(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOE bit</td>
<td>OSSI bit</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

**Note**: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.
42.4.12 TIM counter (TIMx_CNT)(x = 1, 8)
Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UIFCPY: UIF copy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.</td>
</tr>
</tbody>
</table>

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 CNT[15:0]: Counter value

42.4.13 TIMx prescaler (TIMx_PSC)(x = 1, 8)
Address offset: 0x28
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>PSC[15:0]: Prescaler value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).</td>
</tr>
<tr>
<td></td>
<td>PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in &quot;reset mode&quot;).</td>
</tr>
</tbody>
</table>

42.4.14 TIMx auto-reload register (TIMx_ARR)(x = 1, 8)
Address offset: 0x2C
Reset value: 0xFFFF

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>ARR[15:0]: Auto-reload value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ARR is the value to be loaded in the actual auto-reload register.</td>
</tr>
<tr>
<td></td>
<td>Refer to the Section 42.3.1: Time-base unit on page 1441 for more details about ARR update and behavior.</td>
</tr>
<tr>
<td></td>
<td>The counter is blocked while the auto-reload value is null.</td>
</tr>
</tbody>
</table>
42.4.15 TIMx repetition counter register (TIMx_RCR)(x = 1, 8)

Address offset: 0x30
Reset value: 0x0000

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

Bits 15:0 REP[15:0]: Repetition counter value
These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.
Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.
It means in PWM mode (REP+1) corresponds to:
the number of PWM periods in edge-aligned mode
the number of half PWM period in center-aligned mode.

42.4.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8)

Address offset: 0x34
Reset value: 0x0000

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CCR1[15:0]</td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:0 CCR1[15:0]: Capture/Compare 1 value
If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.
### 42.4.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8)

Address offset: 0x38
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>CCR2[15:0]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 15:0 CCR2[15:0]: Capture/Compare value**

- **If channel CC2 is configured as output:**
  CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
  It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
  The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output.

- **If channel CC2 is configured as input:**
  CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

### 42.4.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8)

Address offset: 0x3C
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>CCR3[15:0]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 15:0 CCR3[15:0]: Capture/Compare value**

- **If channel CC3 is configured as output:**
  CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
  It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
  The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC3 output.

- **If channel CC3 is configured as input:**
  CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.
42.4.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8)

Address offset: 0x40
Reset value: 0x0000

<table>
<thead>
<tr>
<th>CCR4[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:0 CCR4[15:0]: Capture/Compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

42.4.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)

Address offset: 0x44
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Res</td>
<td>Res</td>
<td>BK2BID</td>
<td>BKBID</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Note: As the bits BK2BID, BKBID, BK2DSRM, BK2DSRM, BK2P, BK2E, BK2F[3:0], BK2F[3:0], AOE, BKP, BKE, OSSR, OSSI, and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

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

Bit 29 BK2BID: Break2 bidirectional
Refer to BKBID description
Bit 28 **BkBID**: Break Bidirectional

- 0: Break input BRK in input mode
- 1: Break input BRK in bidirectional mode

In the bidirectional mode (BkBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

*Note:* This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 Disarm

Refer to BKDSRM description

Bit 26 **BKDSRM**: Break Disarm

- 0: Break input BRK is armed
- 1: Break input BRK is disarmed

This bit is cleared by hardware when no break source is active.

The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity

- 0: Break input BRK2 is active low
- 1: Break input BRK2 is active high

*Note:* This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable

This bit enables the complete break 2 protection (including all sources connected to bk_acth and BKIN sources, as per Figure 356: Break and Break2 circuitry overview).

- 0: Break2 function disabled
- 1: Break2 function enabled

*Note:* The BKIN2 must only be used with OSSR = OSSI = 1.

*Note:* This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bit-field defines the frequency used to sample BRK2 input and the length of the digital filter applied to BRK2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, BRK2 acts asynchronously
- 0001: \(f_{\text{SAMPLING}}=f_{\text{CK_INT}}\), \(N=2\)
- 0010: \(f_{\text{SAMPLING}}=f_{\text{CK_INT}}\), \(N=4\)
- 0011: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/2\), \(N=8\)
- 0100: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/2\), \(N=6\)
- 0101: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/2\), \(N=8\)
- 0110: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/4\), \(N=6\)
- 0111: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/4\), \(N=8\)
- 1000: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/8\), \(N=6\)
- 1001: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/8\), \(N=8\)
- 1010: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/16\), \(N=5\)
- 1011: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/16\), \(N=6\)
- 1100: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/16\), \(N=8\)
- 1101: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/32\), \(N=5\)
- 1110: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/32\), \(N=6\)
- 1111: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/32\), \(N=8\)

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM\(x\)_BDTR register).*

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, BRK acts asynchronously
- 0001: \(f_{\text{SAMPLING}}=f_{\text{CK_INT}}\), \(N=2\)
- 0010: \(f_{\text{SAMPLING}}=f_{\text{CK_INT}}\), \(N=4\)
- 0011: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/2\), \(N=8\)
- 0100: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/2\), \(N=6\)
- 0101: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/2\), \(N=8\)
- 0110: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/4\), \(N=6\)
- 0111: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/4\), \(N=8\)
- 1000: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/8\), \(N=6\)
- 1001: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/8\), \(N=8\)
- 1010: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/16\), \(N=5\)
- 1011: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/16\), \(N=6\)
- 1100: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/16\), \(N=8\)
- 1101: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/32\), \(N=5\)
- 1110: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/32\), \(N=6\)
- 1111: \(f_{\text{SAMPLING}}=f_{\text{DTS}}/32\), \(N=8\)

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM\(x\)_BDTR register).*
Bit 15 **MOE**: Main output enable
This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (BRK or BRK2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.
- 0: In response to a break 2 event. OC and OCN outputs are disabled
- 1: In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

See OC/OCN enable description for more details (Section 42.4.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)).

Bit 14 **AOE**: Automatic output enable
- 0: MOE can be set only by software
- 1: MOE can be set by software or automatically at the next update event (if none of the break inputs BRK and BRK2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity
- 0: Break input BRK is active low
- 1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable
This bit enables the complete break protection (including all sources connected to bk_acth and BKIN sources, as per Figure 356: Break and Break2 circuitry overview).
- 0: Break function disabled
- 1: Break function enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode
This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (Section 42.4.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)).
- 0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).
- 1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 10 **OSSI**: Off-state selection for Idle mode
This bit is used when MOE=0 due to a break event or by a software write, on channels configured as outputs.
See OC/OCN enable description for more details (Section 42.4.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).
1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

**Note:** This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration
These bits offer a write protection against software errors.
00: LOCK OFF - No bit is write protected.
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BK2BID, BK2SID, BK2DSRM, BK2DSRM, BK2P, BK2E, BK2F[3:0], BK2F[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits in TIMx_BDTR register can no longer be written.
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

**Note:** The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup
This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x t_dig with t_dig=tDTS.
DTG[7:5]=10x => DT=(64+DTG[5:0])x t_dig with t_dig=2x tDTS.
DTG[7:5]=110 => DT=(32+DTG[4:0])x t_dig with t_dig=8x tDTS.
DTG[7:5]=111 => DT=(32+DTG[4:0])x t_dig with t_dig=16x tDTS.

Example if tDTS=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 us to 31750 ns by 250 ns steps,
32 us to 63us by 1 us steps,
64 us to 126 us by 2 us steps

**Note:** This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

### 42.4.21 TIMx DMA control register (TIMx_DCR)(x = 1, 8)

Address offset: 0x48
Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.
Bits 12:8  **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

- 00000: 1 transfer
- 00001: 2 transfers
- 00010: 3 transfers
- ...
- 10001: 18 transfers

**Example**: Let us consider the following transfer: DBL = 7 bytes & DBA = TIMx_CR1.

- If DBL = 7 bytes and DBA = TIMx_CR1 represents the address of the byte to be transferred, the address of the transfer should be given by the following equation: (TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data is copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

- If the DMA Data Size is configured in half-words, 16-bit data is transferred to each of the 7 registers.
- If the DMA Data Size is configured in bytes, the data is also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on.

So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5  **Reserved**, must be kept at reset value.

Bits 4:0  **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

**Example**:

- 00000: TIMx_CR1,
- 00001: TIMx_CR2,
- 00010: TIMx_SMCR,
- ...

### 42.4.22  TIMx DMA address for full transfer

**(TIMx_DMAR)(x = 1, 8)**

**Address offset**: 0x4C

**Reset value**: 0x0000 0000

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

**DMAB[31:16]**

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

**DMAB[15:0]**
42.4.23 TIMx capture/compare mode register 3
(TIMx_CCMR3)(x = 1, 8)

Address offset: 0x54
Reset value: 0x0000 0000
The channels 5 and 6 can only be configured in output.

Output compare mode:

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

Bits 31:0 DMAB[31:0]: DMA register for burst accesses
A read or write operation to the DMAR register accesses the register located at the address
(TIMx_CR1 address) + (DBA + DMA index) x 4
where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base
address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA
transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

Bits 31:25 Reserved, must be kept at reset value.
Bits 23:17 Reserved, must be kept at reset value.
Bit 15 OC6CE: Output compare 6 clear enable
Refer to OC1CE description.

Bits 24, 14, 13, 12 OC6M[3:0]: Output compare 6 mode
Refer to OC1M description.

Bit 11 OC6PE: Output compare 6 preload enable
Refer to OC1PE description.

Bit 10 OC6FE: Output compare 6 fast enable
Refer to OC1FE description.

Bits 9:8 Reserved, must be kept at reset value.
Bit 7 OC5CE: Output compare 5 clear enable
Refer to OC1CE description.

Bits 16, 6, 5, 4 OC5M[3:0]: Output compare 5 mode
Refer to OC1M description.

Bit 3 OC5PE: Output compare 5 preload enable
Refer to OC1PE description.

Bit 2 OC5FE: Output compare 5 fast enable
Refer to OC1FE description.

Bits 1:0 Reserved, must be kept at reset value.
42.4.24 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8)

Address offset: 0x58

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>GC5C3: Group Channel 5 and Channel 3</td>
<td>Group Channel 5 and Channel 3</td>
<td>Distortion on Channel 3 output: 0: No effect of OC5REF on OC3REFC 1: OC3REFC is the logical AND of OC3REFC and OC5REF This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2). Note: it is also possible to apply this distortion on combined PWM signals.</td>
</tr>
<tr>
<td>30</td>
<td>GC5C2: Group Channel 5 and Channel 2</td>
<td>Group Channel 5 and Channel 2</td>
<td>Distortion on Channel 2 output: 0: No effect of OC5REF on OC2REFC 1: OC2REFC is the logical AND of OC2REFC and OC5REF This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1). Note: it is also possible to apply this distortion on combined PWM signals.</td>
</tr>
<tr>
<td>29</td>
<td>GC5C1: Group Channel 5 and Channel 1</td>
<td>Group Channel 5 and Channel 1</td>
<td>Distortion on Channel 1 output: 0: No effect of OC5REF on OC1REFC5 1: OC1REFC is the logical AND of OC1REFC and OC5REF This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1). Note: it is also possible to apply this distortion on combined PWM signals.</td>
</tr>
<tr>
<td>28:16</td>
<td>Reserved</td>
<td>Reserved</td>
<td>must be kept at reset value.</td>
</tr>
<tr>
<td>15:0</td>
<td>CCR5[15:0]: Capture/Compare 5 value</td>
<td>Capture/Compare 5 value</td>
<td>CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC5 output.</td>
</tr>
</tbody>
</table>
### 42.4.25 TIMx capture/compare register 6 (TIMx_CCR6) (x = 1, 8)

Address offset: 0x5C  
Reset value: 0x0000

| Bits 15:0 | CCR6[15:0]: Capture/Compare 6 value  
|-----------|-----------------------------------|
| CCR6[15:0]: Capture/Compare 6 value  
| It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.  
| The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC6 output. |

### 42.4.26 TIM1 alternate function option register 1 (TIM1_AF1)

Address offset: 0x60  
Reset value: 0x0000 0001

| Bits 31:18 | Reserved, must be kept at reset value.  
|------------|-----------------------------------|
| ETRSEL[3:0]: ETR source selection  
| These bits select the ETR input source.  
| 0000: ETR input is connected to I/O  
| 0001: COMP1 output  
| 0010: COMP2 output  
| 0011: ADC1 AWD1  
| 0100: ADC1 AWD2  
| 0101: ADC1 AWD3  
| Others: Reserved  
| Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register) |

| Bits 13:12 | Reserved, must be kept at reset value.  
|------------|-----------------------------------|
Bit 11 **BKCMP2P**: BRK COMP2 input polarity
This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP2 input is active high
1: COMP2 input is active low

_Note:_ This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP1 input is active high
1: COMP1 input is active low

_Note:_ This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity
This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: BKIN input is active high
1: BKIN input is active low

_Note:_ This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK0E**: BRK dfsdm1_break[0] enable
This bit enables the dfsdm1_break[0] for the timer’s BRK input. dfsdm1_break[0] output is ‘ORed’ with the other BRK sources.
0: dfsdm1_break[0] input disabled
1: dfsdm1_break[0] input enabled

_Note:_ This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 2 **BKCMP2E**: BRK COMP2 enable
This bit enables the COMP2 for the timer’s BRK input. COMP2 output is ‘ORed’ with the other BRK sources.
0: COMP2 input disabled
1: COMP2 input enabled

_Note:_ This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable
This bit enables the COMP1 for the timer’s BRK input. COMP1 output is ‘ORed’ with the other BRK sources.
0: COMP1 input disabled
1: COMP1 input enabled

_Note:_ This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled
1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

**Note:** Refer to Figure 335: TIM1/TIM8 ETR input circuitry and to Figure 356: Break and Break2 circuitry overview.

### 42.4.27 TIM1 Alternate function register 2 (TIM1_AF2)

**Address offset:** 0x64

**Reset value:** 0x0000 0001

<table>
<thead>
<tr>
<th>Bit 31-12 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 11 <strong>BK2CMP2P</strong>: BRK2 COMP2 input polarity</td>
</tr>
<tr>
<td>This bit selects the COMP2 input sensitivity. It must be programmed together with the BK2P polarity bit.</td>
</tr>
<tr>
<td>0: COMP2 input is active low</td>
</tr>
<tr>
<td>1: COMP2 input is active high</td>
</tr>
<tr>
<td><em>Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</em></td>
</tr>
<tr>
<td>Bit 10 <strong>BK2CMP1P</strong>: BRK2 COMP1 input polarity</td>
</tr>
<tr>
<td>This bit selects the COMP1 input sensitivity. It must be programmed together with the BK2P polarity bit.</td>
</tr>
<tr>
<td>0: COMP1 input is active low</td>
</tr>
<tr>
<td>1: COMP1 input is active high</td>
</tr>
<tr>
<td><em>Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</em></td>
</tr>
<tr>
<td>Bit 9 <strong>BK2INP</strong>: BRK2 BKIN2 input polarity</td>
</tr>
<tr>
<td>This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.</td>
</tr>
<tr>
<td>0: BKIN2 input is active low</td>
</tr>
<tr>
<td>1: BKIN2 input is active high</td>
</tr>
<tr>
<td><em>Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</em></td>
</tr>
</tbody>
</table>
42.4.28 TIM8 Alternate function option register 1 (TIM8_AF1)

Address offset: 0x60
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ETRSEL[3:2]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw rw</td>
</tr>
</tbody>
</table>

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

Bits 31:18 Reserved, must be kept at reset value.

Note: Refer to Figure 356: Break and Break2 circuitry overview.
Bits 17:14 **ETRSEL[3:0]**: ETR source selection
   These bits select the ETR input source.
   0000: ETR input is connected to I/O
   0001: COMP1 output
   0010: COMP2 output
   0011: ADC2 AWD1
   0100: ADC2 AWD2
   0101: ADC2 AWD3
   Others: Reserved
   Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity
   This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
   0: COMP2 input is active high
   1: COMP2 input is active low
   Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity
   This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
   0: COMP1 input is active high
   1: COMP1 input is active low
   Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity
   This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
   0: BKIN input is active high
   1: BKIN input is active low
   Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK2E**: BRK dfsdm1_break[2] enable
   This bit enables the dfsdm1_break[2] for the timer’s BRK input. dfsdm1_break[2] output is ‘ORed’ with the other BRK sources.
   0: dfsdm1_break[2] input disabled
   1: dfsdm1_break[2] input enabled
   Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 2 **BKCMP2E**: BRK COMP2 enable
   This bit enables the COMP2 for the timer’s BRK input. COMP2 output is ‘ORed’ with the other BRK sources.
   0: COMP2 input disabled
   1: COMP2 input enabled
   Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
42.4.29 TIM8 Alternate function option register 2 (TIM8_AF2)

Address offset: 0x64
Reset value: 0x0000 0001

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BK2CMP2P**: BRK2 COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BK2P polarity bit.
0: COMP2 input is active low
1: COMP2 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 10 **BK2CMP1P**: BRK2 COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BK2P polarity bit.
0: COMP1 input is active low
1: COMP1 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*
Bit 9 **BK2INP**: BRK2 BKIN2 input polarity  
This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.
- 0: BKIN2 input is active low
- 1: BKIN2 input is active high

*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2DF1BK3E**: BRK2 dfsdm1_break[3] enable  
This bit enables the dfsdm1_break[3] for the timer’s BRK2 input. dfsdm1_break[3] output is ‘ORed’ with the other BRK2 sources.
- 0: dfsdm1_break[3] input disabled
- 1: dfsdm1_break[3] input enabled

*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 2 **BK2CMP2E**: BRK2 COMP2 enable  
This bit enables the COMP2 for the timer’s BRK2 input. COMP2 output is ‘ORed’ with the other BRK2 sources.
- 0: COMP2 input disabled
- 1: COMP2 input enabled

*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BK2CMP1E**: BRK2 COMP1 enable  
This bit enables the COMP1 for the timer’s BRK2 input. COMP1 output is ‘ORed’ with the other BRK2 sources.
- 0: COMP1 input disabled
- 1: COMP1 input enabled

*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BK2INE**: BRK2 BKIN input enable  
This bit enables the BKIN2 alternate function input for the timer’s BRK2 input. BKIN2 input is ‘ORed’ with the other BRK2 sources.
- 0: BKIN2 input disabled
- 1: BKIN2 input enabled

*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note*: Refer to *Figure 356: Break and Break2 circuitry overview.*
### 42.4.30 TIM1 timer input selection register (TIM1_TISEL)

Address offset: 0x68  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
| 30  | TI4SEL[3:0] | selects TI4[0] to TI4[15] input  
0000: TIM1_CH4 input  
Others: Reserved |
| 29  | Res. | Reserved, must be kept at reset value. |
0000: TIM1_CH3 input  
Others: Reserved |
| 27  | Res. | Reserved, must be kept at reset value. |
| 26  | TI2SEL[3:0] | selects TI2[0] to TI2[15] input  
0000: TIM1_CH2 input  
Others: Reserved |
| 25  | Res. | Reserved, must be kept at reset value. |
| 24  | TI1SEL[3:0] | selects TI1[0] to TI1[15] input  
0000: TIM1_CH1 input  
0001: COMP1 output  
Others: Reserved |

### 42.4.31 TIM8 timer input selection register (TIM8_TISEL)

Address offset: 0x68  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
| 30  | TI4SEL[3:0] | selects TI4[0] to TI4[15] input  
0000: TIM8_CH4 input  
Others: Reserved |
| 29  | Res. | Reserved, must be kept at reset value. |
0000: TIM8_CH3 input  
Others: Reserved |
| 27  | Res. | Reserved, must be kept at reset value. |
| 26  | TI2SEL[3:0] | selects TI2[0] to TI2[15] input  
0000: TIM8_CH2 input  
Others: Reserved |
| 25  | Res. | Reserved, must be kept at reset value. |
| 24  | TI1SEL[3:0] | selects TI1[0] to TI1[15] input  
0000: TIM8_CH1 input  
0001: COMP1 output  
Others: Reserved |
Bits 23:20  Reserved, must be kept at reset value.

            0000: TIM8_CH3 input
            Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8   **T12SEL[3:0]**: selects T12[0] to T12[15] input
            0000: TIM8_CH2 input
            Others: Reserved

Bits 7:4   Reserved, must be kept at reset value.

Bits 3:0   **T11SEL[3:0]**: selects T11[0] to T11[15] input
            0000: TIM8_CH1 input
            0001: COMP2 output
            Others: Reserved
42.4.32 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset values</th>
<th>Reset values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>TIM1_CR1</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>0x04</td>
<td>TIM1_CR2</td>
<td>0x04</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x08</td>
<td>TIM1_SMCR</td>
<td>0x08</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0C</td>
<td>TIM1_DIER</td>
<td>0x0C</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x10</td>
<td>TIM1_SR</td>
<td>0x10</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x14</td>
<td>TIM1_EGR</td>
<td>0x14</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x18</td>
<td>TIM1_CCMR1</td>
<td>0x18</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x1C</td>
<td>TIM1_CCMR2</td>
<td>0x1C</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x20</td>
<td>TIM1_CCER</td>
<td>0x20</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Table 307. TIM1 register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset values</th>
<th>Reset values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>TIM1_CR1</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>0x04</td>
<td>TIM1_CR2</td>
<td>0x04</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x08</td>
<td>TIM1_SMCR</td>
<td>0x08</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0C</td>
<td>TIM1_DIER</td>
<td>0x0C</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x10</td>
<td>TIM1_SR</td>
<td>0x10</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x14</td>
<td>TIM1_EGR</td>
<td>0x14</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x18</td>
<td>TIM1_CCMR1</td>
<td>0x18</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x1C</td>
<td>TIM1_CCMR2</td>
<td>0x1C</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x20</td>
<td>TIM1_CCER</td>
<td>0x20</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>
### Table 307. TIM1 register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x24</td>
<td>TIM1_CNT</td>
<td>CNT[15:0]</td>
<td></td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x28</td>
<td>TIM1_PSC</td>
<td>PSC[15:0]</td>
<td></td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x2C</td>
<td>TIM1_ARR</td>
<td>ARR[15:0]</td>
<td></td>
<td>1111111111111111</td>
</tr>
<tr>
<td>0x30</td>
<td>TIM1_RCR</td>
<td>REP[15:0]</td>
<td></td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x34</td>
<td>TIM1_CCR1</td>
<td>CCR1[15:0]</td>
<td></td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x38</td>
<td>TIM1_CCR2</td>
<td>CCR2[15:0]</td>
<td></td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x3C</td>
<td>TIM1_CCR3</td>
<td>CCR3[15:0]</td>
<td></td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x40</td>
<td>TIM1_CCR4</td>
<td>CCR4[15:0]</td>
<td></td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x44</td>
<td>TIM1_BDTR</td>
<td>BK2F[3:0]</td>
<td></td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x48</td>
<td>TIM1_DCR</td>
<td>DBL[4:0]</td>
<td></td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x4C</td>
<td>TIM1_DMAR</td>
<td>DMAB[31:0]</td>
<td></td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x54</td>
<td>TIM1_CCMR3</td>
<td>OC5M[3]</td>
<td></td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x58</td>
<td>TIM1_CCR5</td>
<td>CCR5[15:0]</td>
<td></td>
<td>0000000000000000</td>
</tr>
</tbody>
</table>
Refer to Section 2.3 on page 128 for the register boundary addresses.

### 42.4.33 TIM8 register map

TIM8 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 308. TIM8 register map and reset values**

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | TIM8_CR1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | TIM8_CR2      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | TIM8_SMCR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | TIM8_DIER     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | TIM8_SR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Table 308. TIM8 register map and reset values (continued)

| Offset | Register name   | Reserved 01  | Reserved 02  | Reserved 03  | Reserved 04  | Reserved 05  | Reserved 06 | Reserved 07 | Reserved 08 | Reserved 09 | Reserved 10 | Reserved 11 | Reserved 12 | Reserved 13 | Reserved 14 | Reserved 15 | Reserved 16 | Reserved 17 | Reserved 18 | Reserved 19 | Reserved 20 | Reserved 21 | Reserved 22 | Reserved 23 | Reserved 24 | Reserved 25 | Reserved 26 |
|--------|----------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| 0x14   | TIM8_EGR       |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |
|        | Reset value    |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |
| 0x18   | TIM8_CCMR1     | OC2M         | OC2CE        | OC2PE        | OC1M         | OC1PE        | OC1FE        | IC2F         | IC2S         | IC1F         | IC1S         | IC1P         | IC1PE        | IC1FE        | IC4F         | IC4S         | IC4P         | IC4PE        | IC4FE        | CC4M         | CC4E         | CC4P         | CC4PE        | CC4FE        |
|        | Output Compare mode |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |
|        | Reset value    | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            |
| 0x1C   | TIM8_CCMR2     | OC4M         | OC4CE        | OC4PE        | OC3M         | OC3PE        | OC3FE        | IC4F         | IC4S         | IC3F         | IC3S         | IC3P         | IC3PE        | IC3FE        | CC6P         | CC6E         | CC5P         | CC5E         | CC4NP        | CC4NE        | CC3NP        | CC3NE        | CC2NP        | CC2NE        |
|        | Output Compare mode |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |
|        | Reset value    | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            |
| 0x20   | TIM8_CCER      |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |
|        | Reset value    | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            |
| 0x24   | TIM8_CNT       |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |
|        | Reset value    | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            |
| 0x28   | TIM8_PSC       |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |
|        | Reset value    | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            |
| 0x2C   | TIM8_ARR       |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |
|        | Reset value    | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            | 1            |
| 0x30   | TIM8_RCR       |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |
|        | Reset value    | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            |
| 0x34   | TIMx_CCR1      |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |
|        | Reset value    | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            |
| 0x38   | TIM8_CCR2      |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |
|        | Reset value    | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            |
| 0x3C   | TIM8_CCR3      |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |
|        | Reset value    | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            |
| 0x40   | TIM8_CCR4      |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |              |
|        | Reset value    | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            | 0            |
Refer to Section 2.3 on page 128 for the register boundary addresses.
43  General-purpose timers (TIM2/TIM3/TIM4/TIM5)

43.1  TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit/32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in Section 43.3.19: Timer synchronization.

43.2  TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3, TIM4) or 32-bit (TIM2 and TIM5) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management
Figure 373. General-purpose timer block diagram

Notes:
- Preload registers transferred to active registers on U event according to control bit
- Event
- Interrupt & DMA output
43.3 **TIM2/TIM3/TIM4/TIM5 functional description**

43.3.1 **Time-base unit**

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:
- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

**Prescaler description**

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 374 and Figure 375* give some examples of the counter behavior when the prescaler ratio is changed on the fly:
Figure 374. Counter timing diagram with prescaler division change from 1 to 2

Figure 375. Counter timing diagram with prescaler division change from 1 to 4
### 43.3.2 Counter modes

#### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

#### Figure 376. Counter timing diagram, internal clock divided by 1

![Counter timing diagram](image)
**Figure 377. Counter timing diagram, internal clock divided by 2**

- CK_PSC
- CNT_EN
- Timer clock = CK_CNT
- Counter register: 0034 0035 0036 0000 0001 0002 0003
- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)

**Figure 378. Counter timing diagram, internal clock divided by 4**

- CK_PSC
- CNT_EN
- Timer clock = CK_CNT
- Counter register: 0035 0036 0000 0001
- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)
Figure 379. Counter timing diagram, internal clock divided by N

Figure 380. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)
Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An Update event can be generate at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn’t change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.
The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

**Figure 382. Counter timing diagram, internal clock divided by 1**

**Figure 383. Counter timing diagram, internal clock divided by 2**
Figure 384. Counter timing diagram, internal clock divided by 4

Figure 385. Counter timing diagram, internal clock divided by N
Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to ‘00’. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

**Figure 387. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6**

1. Here, center-aligned mode 1 is used (for more details refer to Section 43.4.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 5) on page 1569).
Figure 388. Counter timing diagram, internal clock divided by 2

Figure 389. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

Note: Here, center_aligned mode 2 or 3 is updated with an UIF on overflow.

1. Center-aligned mode 2 or 3 is used with an UIF on overflow.
Figure 390. Counter timing diagram, internal clock divided by N

Figure 391. Counter timing diagram, Update event with ARPE=1 (counter underflow)
43.3.3 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin (TiX)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRX): using one timer as prescaler for another timer, for example, Timer X can be configured to act as a prescaler for Timer Y. Refer to : Using one timer as prescaler for another timer on page 1584 for more details.

**Internal clock source (CK_INT)**

If the slave mode controller is disabled (SMS=000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

*Figure 393* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.
External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

1. Codes ranging from 01000 to 11111: ITRy.

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S= ‘01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
**Note:** The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 and CC2NP=0 in the TIMx_CCER register.

5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.

6. Select TI2 as the input source by writing TS=00110 in the TIMx_SMCR register.

7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

**Figure 395. Control circuit in external clock mode 1**

External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

*Figure 396* gives an overview of the external trigger input block.
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. Select the proper ETR source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
2. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register.
4. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register.
5. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most $\frac{1}{4}$ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETPS prescaler setting.
43.3.4 Capture/Compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding Tlx input to generate a filtered signal TlxF. Then, an edge detector with polarity selection generates a signal (TlxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.
The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.
43.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let’s imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.
43.3.6 PWM input mode

This mode is a particular case of input capture mode. The procedure is the same except:
- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper Ti1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to ‘0’ and the CC1NP bit to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to ‘1’ and the CC2NP bit to ‘0’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.
43.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (ocxref/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

43.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCRER register). The output pin can keep its level (OCxM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, one must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.
The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 402.

**Figure 402. Output compare mode, toggle on OC1**

```plaintext
<table>
<thead>
<tr>
<th>TIM1_CNT</th>
<th>0039</th>
<th>003A</th>
<th>003B</th>
<th>B200</th>
<th>B201</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM1_CCR1</td>
<td></td>
<td></td>
<td></td>
<td>003A</td>
<td></td>
</tr>
<tr>
<td>OC1REF= OC1</td>
<td></td>
<td></td>
<td></td>
<td>B201</td>
<td></td>
</tr>
</tbody>
</table>
```

**43.3.9 PWM mode**

Pulse width modulation mode allows to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or '111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). However, to comply with the OCREF_CLR functionality (OCREF can be...
cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison or
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM='000) to one of the PWM modes (OCxM='110 or ‘111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

**PWM edge-aligned mode**

**Upcounting configuration**

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode on page 1549*.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1. If the compare value is 0 then OCxREF is held at ‘0. *Figure 403* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

*Figure 403. Edge-aligned PWM waveforms (ARR=8)*
**Downcounting configuration**

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to *Downcounting mode on page 1552*.

In PWM mode 1, the reference signal ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then ocxref is held at 100%. PWM is not possible in this mode.

**PWM center-aligned mode**

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from '00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting) on page 1555*.

*Figure 404* shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.
Figure 404. Center-aligned PWM waveforms (ARR=8)

Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.

- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.
43.3.10 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

**Note:** The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 2.

*Figure 405* shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1).

![Figure 405. Generation of 2 phase-shifted PWM signals with 50% duty cycle](image)

43.3.11 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

**Note:** The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

*Figure 406* shows an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1

![Figure 406. Combined PWM mode on channels 1 and 3](image)

**43.3.12 Clearing the OCxREF signal on an external event**

The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The ocref_clr_int is connected to the ETRF signal (ETR after filtering).
The OCxREF signal for a given channel can be reset by applying a high level on the ETRF input (OCxCE enable bit set to 1 in the corresponding TIMx_CCMRx register). OCxREF remains low until the next update event (UEV) occurs.

This function can be used only in the output compare and PWM modes. It does not work in forced mode.

For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application’s needs.

*Figure 407* shows the behavior of the OCxREF signal when the ETRF input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.
### One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- $\text{CNT}<\text{CCR}x \leq \text{ARR}$ (in particular, $0<\text{CCR}x$),

**Figure 408. Example of one-pulse mode.**

For example one may want to generate a positive pulse on OC1 with a length of $t_{\text{PULSE}}$ and after a delay of $t_{\text{DELAY}}$ as soon as a positive edge is detected on the TI2 input pin.

Let’s use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 on TI2 by writing CC2S=01 in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to ‘110 in the TIMx_SMCR register (trigger mode).
The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The $t_{\text{DELAY}}$ is defined by the value written in the TIMx_CCR1 register.
- The $t_{\text{PULSE}}$ is defined by the difference between the auto-reload value and the compare value ($\text{TIMx.ARR - TIMx.CCR1}$).
- Let’s say one wants to build a waveform with a transition from ‘0 to ‘1 when a compare match occurs and a transition from ‘1 to ‘0 when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0 in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low. Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to ‘0’, so the Repetitive Mode is selected.

**Particular case: OCx fast enable:**

In One-pulse mode, the edge detection on Tlx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay $t_{\text{DELAY min}}$ we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 43.3.14 Retriggerable one pulse mode (OPM)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in Section 43.3.13:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

**Note:** In retriggerable one pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.
43.3.15 Encoder interface mode

To select Encoder Interface mode write SMS='001 in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to Table 309. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to ‘1’). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder’s position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.
An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder’s differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

Figure 410 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S = 01 (TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S = 01 (TIMx_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P and CC1NP = ‘0’ (TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P and CC2NP = ‘0’ (TIMx_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS = 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN = 1 (TIMx_CR1 register, Counter is enabled)

### Table 309. Counting direction versus encoder signals

<table>
<thead>
<tr>
<th>Active edge</th>
<th>Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)</th>
<th>TI1FP1 signal</th>
<th>TI2FP2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rising</td>
<td>Falling</td>
<td>Rising</td>
</tr>
<tr>
<td>Counting on TI1 only</td>
<td>High</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>Down</td>
</tr>
<tr>
<td>Counting on TI2 only</td>
<td>High</td>
<td>No Count</td>
<td>No Count</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>No Count</td>
<td>No Count</td>
</tr>
<tr>
<td>Counting on TI1 and TI2</td>
<td>High</td>
<td>Down</td>
<td>Up</td>
</tr>
</tbody>
</table>

Figure 411 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).
The timer, when configured in Encoder Interface mode provides information on the sensor’s current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

43.3.16 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register’s bit 31 (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter’s most significant bit is only accessible in write mode).

43.3.17 Timer input XOR function

The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1 to TIMx_CH3. The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in Section 42.3.25: Interfacing with Hall sensors on page 1490.
43.3.18 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

**Slave mode: Reset mode**

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).

2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

![Figure 412. Control circuit in reset mode](image)

**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:
1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCRER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Note: The configuration “CCxP=CCxNP=1” (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

**Slave mode: Trigger mode**

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write
CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

**Figure 414. Control circuit in trigger mode**

![Control circuit in trigger mode](MS31403V1)

**Slave mode: External Clock mode 2 + trigger mode**

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.

2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=011 in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edge only).

3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

**Figure 415. Control circuit in external clock mode 2 + trigger mode**

![Control circuit in external clock mode 2 + trigger mode](image)

### 43.3.19 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

*Figure 416: Master/Slave timer example* presents an overview of the trigger selection and the master mode selection blocks.

**Figure 416. Master/Slave timer example**

![Master/Slave timer example](image)
Using one timer as prescaler for another timer

For example, TIM3 can be configured to act as a prescaler for TIM2. Refer to Figure 416. To do this:

1. Configure TIM3 in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM3_CR2 register, a rising edge is output on TRGO each time an update event is generated.

2. To connect the TRGO output of TIM3 to TIM2, TIM2 must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM2_SMCR register (writing TS=00010).

3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM3 trigger signal (which correspond to the TIM3 counter overflow).

4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If OCx is selected on TIM3 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 3. Refer to Figure 416 for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM3 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT (f_{CK_CNT} = f_{CK_INT}/3).

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).

2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).

3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).

4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).

5. Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).

6. Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.

Figure 417. Gating TIM2 with OC1REF of TIM3
In the example in *Figure 417*, the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM3. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 418*), we synchronize TIM3 and TIM2. TIM3 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM3 is disabled by writing ‘0’ to the CEN bit in the TIM3_CR1 register:

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Reset TIM3 by writing ‘1’ in UG bit (TIM3_EGR register).
6. Reset TIM2 by writing ‘1’ in UG bit (TIM2_EGR register).
7. Initialize TIM2 to 0xE7 by writing ‘0xE7’ in the TIM2_CNT register.
8. Enable TIM2 by writing ‘1’ in the CEN bit (TIM2_CR1 register).
9. Start TIM3 by writing ‘1’ in the CEN bit (TIM3_CR1 register).
10. Stop TIM3 by writing ‘0’ in the CEN bit (TIM3_CR1 register).

*Figure 418. Gating TIM2 with Enable of TIM3*

<table>
<thead>
<tr>
<th>CK_INT</th>
<th>TIM3-CEN=CNT_EN</th>
<th>TIM3-CNT_INIT</th>
<th>TIM3-CNT</th>
<th>TIM2-CNT</th>
<th>TIM2-CNT_INIT</th>
<th>TIM2-write CNT</th>
<th>TIM2-TIF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>75</td>
<td>00</td>
<td>01</td>
<td>02</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E7</td>
<td>E8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write TIF = 0</td>
<td></td>
</tr>
</tbody>
</table>

**Using one timer to start another timer**

In this example, we set the enable of Timer 2 with the update event of Timer 3. Refer to *Figure 416* for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write ‘0’ to the CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT (\(f_{\text{CK_CNT}} = f_{\text{CK_INT}}/3\)).
1. Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3_CR2 register).
2. Configure the TIM3 period (TIM3_ARR registers).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
5. Start TIM3 by writing ‘1’ in the CEN bit (TIM3_CR1 register).

**Figure 419. Triggering TIM2 with update of TIM3**

As in the previous example, both counters can be initialized before starting counting. **Figure 420** shows the behavior with the same configuration as in **Figure 419** but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

**Figure 420. Triggering TIM2 with Enable of TIM3**

**Starting 2 timers synchronously in response to an external trigger**

In this example, we set the enable of TIM3 when its T1 input rises, and the enable of TIM2 with the enable of TIM3. Refer to **Figure 416** for connections. To ensure the counters are
aligned, TIM3 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIM2):

1. Configure TIM3 master mode to send its Enable as trigger output (MMS=001 in the TIM3_CR2 register).

2. Configure TIM3 slave mode to get the input trigger from TI1 (TS=00100 in the TIM3_SMCR register).

3. Configure TIM3 in trigger mode (SMS=110 in the TIM3_SMCR register).

4. Configure the TIM3 in Master/Slave mode by writing MSM=1 (TIM3_SMCR register).

5. Configure TIM2 to get the input trigger from TIM3 (TS=00000 in the TIM2_SMCR register).

6. Configure TIM2 in trigger mode (SMS=110 in the TIM2_SMCR register).

When a rising edge occurs on TI1 (TIM3), both counters starts counting synchronously on the internal clock and both TIF flags are set.

**Note:** In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on TIM3.

**Figure 421. Triggering TIM3 and TIM2 with TIM3 TI1 input**

<table>
<thead>
<tr>
<th>CK_INT</th>
<th>TIM3-TI1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM3-CEN=CNT_EN</td>
<td></td>
</tr>
<tr>
<td>TIM3-CK_PSC</td>
<td></td>
</tr>
<tr>
<td>TIM3-CNT</td>
<td>00</td>
</tr>
<tr>
<td>TIM3-TIF</td>
<td></td>
</tr>
<tr>
<td>TIM2-CEN=CNT_EN</td>
<td></td>
</tr>
<tr>
<td>TIM2-CK_PSC</td>
<td></td>
</tr>
<tr>
<td>TIM2-CNT</td>
<td>00</td>
</tr>
<tr>
<td>TIM2-TIF</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

### 43.3.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.
The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:
   - DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

43.3.21 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter either continues to work normally or stops, depending on TIMx configuration bit in DBGMCU module. For more details, refer to Section 60.5.8: Microcontroller debug unit (DBGMCU).

For safety purposes, when the counter is stopped (TIMx = 1 in DBGMCU_APB1FZ2), the outputs are disabled.
43.4 **TIM2/TIM3/TIM4/TIM5 registers**

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

### 43.4.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5)

Address offset: 0x00

Reset value: 0x0000

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

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP:** UIF status bit remapping
- 0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
- 1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]:** Clock division
- 00: \( t_{DTS} = t_{CK\_INT} \)
- 01: \( t_{DTS} = 2 \times t_{CK\_INT} \)
- 10: \( t_{DTS} = 4 \times t_{CK\_INT} \)
- 11: Reserved

Bit 7 **ARPE:** Auto-reload preload enable
- 0: TIMx ARR register is not buffered
- 1: TIMx ARR register is buffered

Bits 6:5 **CMS[1:0]:** Center-aligned mode selection
- 00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
- 01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
- 10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
- 11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

*Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)*

Bit 4 **DIR:** Direction
- 0: Counter used as upcounter
- 1: Counter used as downcounter

*Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.*
Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled.
These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled

*Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware. CEN is cleared automatically in one-pulse mode, when an update event occurs.*

### 43.4.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5)

Address offset: 0x04
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T11S</td>
<td>MMS[2:0]</td>
<td>CCDS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table above shows the bits as being read-only (rw).
Bits 15:8  Reserved, must be kept at reset value.

Bit 7  TI1S: TI1 selection
0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)
See also Section 42.3.25: Interfacing with Hall sensors on page 1490

Bit 6:4  MMS[2:0]: Master mode selection
These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: Reset - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: Enable - the Counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: Update - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: Compare Pulse - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred.
100: Compare - OC1REF signal is used as trigger output (TRGO)
101: Compare - OC2REF signal is used as trigger output (TRGO)
110: Compare - OC3REF signal is used as trigger output (TRGO)
111: Compare - OC4REF signal is used as trigger output (TRGO)

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3  CCDS: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bits 2:0  Reserved, must be kept at reset value.
### 43.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5)

Address offset: 0x08
Reset value: 0x0000 0000

| Bit 31:22 | Reserved, must be kept at reset value. |
| Bit 19:17 | Reserved, must be kept at reset value. |

**Bit 15** **ETP**: External trigger polarity
This bit selects whether ETR or ETR is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge
1: ETR is inverted, active at low level or falling edge

**Bit 14** **ECE**: External clock enable
This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

**Note**: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

**Bits 13:12** **ETPS[1:0]**: External trigger prescaler
External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.
00: Prescaler OFF
01: ETRP frequency divided by 2
10: ETRP frequency divided by 4
11: ETRP frequency divided by 8

<table>
<thead>
<tr>
<th>Address offset: 0x08</th>
<th>Reset value: 0x0000 0000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:22 Reserved, must be kept at reset value.</td>
<td>Bits 19:17 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 15 ETP: External trigger polarity</td>
<td>Bit 14 ECE: External clock enable</td>
</tr>
<tr>
<td>This bit selects whether ETR or ETR is used for trigger operations</td>
<td>This bit enables External clock mode 2.</td>
</tr>
<tr>
<td>0: ETR is non-inverted, active at high level or rising edge</td>
<td>0: External clock mode 2 disabled</td>
</tr>
<tr>
<td>1: ETR is inverted, active at low level or falling edge</td>
<td>1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.</td>
</tr>
<tr>
<td><strong>Note</strong>: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).</td>
<td><strong>Note</strong>: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).</td>
</tr>
<tr>
<td>It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).</td>
<td>If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.</td>
</tr>
<tr>
<td><strong>Bits 13:12</strong> <strong>ETPS[1:0]</strong>: External trigger prescaler</td>
<td><strong>Bits 13:12</strong> <strong>ETPS[1:0]</strong>: External trigger prescaler</td>
</tr>
<tr>
<td>External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.</td>
<td>External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.</td>
</tr>
<tr>
<td>00: Prescaler OFF</td>
<td>00: Prescaler OFF</td>
</tr>
<tr>
<td>01: ETRP frequency divided by 2</td>
<td>01: ETRP frequency divided by 2</td>
</tr>
<tr>
<td>10: ETRP frequency divided by 4</td>
<td>10: ETRP frequency divided by 4</td>
</tr>
<tr>
<td>11: ETRP frequency divided by 8</td>
<td>11: ETRP frequency divided by 8</td>
</tr>
</tbody>
</table>
Bits 11:8 **ETF[3:0]:** External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- **0000:** No filter, sampling is done at $f_{DTS}$
- **0001:** $f_{SAMPLING}=f_{CK\_INT}$, $N=2$
- **0010:** $f_{SAMPLING}=f_{CK\_INT}$, $N=4$
- **0011:** $f_{SAMPLING}=f_{CK\_INT}$, $N=8$
- **0100:** $f_{SAMPLING}=f_{DTS}/2$, $N=6$
- **0101:** $f_{SAMPLING}=f_{DTS}/2$, $N=8$
- **0110:** $f_{SAMPLING}=f_{DTS}/4$, $N=6$
- **0111:** $f_{SAMPLING}=f_{DTS}/4$, $N=8$
- **1000:** $f_{SAMPLING}=f_{DTS}/8$, $N=6$
- **1001:** $f_{SAMPLING}=f_{DTS}/8$, $N=8$
- **1010:** $f_{SAMPLING}=f_{DTS}/16$, $N=5$
- **1011:** $f_{SAMPLING}=f_{DTS}/16$, $N=6$
- **1100:** $f_{SAMPLING}=f_{DTS}/16$, $N=8$
- **1101:** $f_{SAMPLING}=f_{DTS}/32$, $N=5$
- **1110:** $f_{SAMPLING}=f_{DTS}/32$, $N=6$
- **1111:** $f_{SAMPLING}=f_{DTS}/32$, $N=8$

Bit 7 **MSM:** Master/Slave mode

- **0:** No action
- **1:** The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.
Bits 21, 20, 6, 5, 4  **TS[4:0]: Trigger selection**

This bit-field selects the trigger input to be used to synchronize the counter.

- 00000: Internal Trigger 0 (ITR0)
- 00001: Internal Trigger 1 (ITR1)
- 00010: Internal Trigger 2 (ITR2)
- 00011: Internal Trigger 3 (ITR3)
- 00100: TI1 Edge Detector (TI1F_ED)
- 00101: Filtered Timer Input 1 (TI1FP1)
- 00110: Filtered Timer Input 2 (TI2FP2)
- 00111: External Trigger input (ETRF)
- 01000: Internal Trigger 4 (ITR4)
- 01001: Internal Trigger 5 (ITR5)
- 01010: Internal Trigger 6 (ITR6)
- 01011: Internal Trigger 7 (ITR7)
- 01100: Internal Trigger 8 (ITR8)
- Others: Reserved

See Table 310: TIMx internal trigger connection on page 1595 for more details on ITRx meaning for each Timer.

**Note:**  These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3  **Reserved, must be kept at reset value.**

Bits 16, 2, 1, 0  **SMS[3:0]: Slave mode selection**

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

- 0000: Slave mode disabled - if CEN = ’1 then the prescaler is clocked directly by the internal clock.
- 0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
- 0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
- 0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
- 0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
- 0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
- 0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
- 0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
- 1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

**Note:**  The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=00100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

**Note:** The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.
### Table 310. TIMx internal trigger connection

<table>
<thead>
<tr>
<th>Slave TIM</th>
<th>ITR0</th>
<th>ITR1</th>
<th>ITR2</th>
<th>ITR3</th>
<th>ITR4</th>
<th>ITR5</th>
<th>ITR6</th>
<th>ITR7</th>
<th>ITR8</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM2</td>
<td>TIM1</td>
<td>TIM8</td>
<td>TIM3</td>
<td>TIM4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIM3</td>
<td>TIM1</td>
<td>TIM2</td>
<td>TIM15</td>
<td>TIM4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIM4</td>
<td>TIM1</td>
<td>TIM2</td>
<td>TIM3</td>
<td>TIM8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIM5</td>
<td>TIM1</td>
<td>TIM8</td>
<td>TIM3</td>
<td>TIM4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**USB1**

**OTG_HS**

**SOF**

**fdcan1_soc**

**USB1**

**OTG_HS**

**SOF**

#### 43.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)

Address offset: 0x0C

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved, must be kept at reset value.</td>
<td>TDE: Trigger DMA request enable</td>
<td>0: Trigger DMA request disabled.</td>
<td>1: Trigger DMA request enabled.</td>
<td>Reserved, must be kept at reset value.</td>
<td>CC4DE: Capture/Compare 4 DMA request enable</td>
<td>0: CC4 DMA request disabled.</td>
<td>1: CC4 DMA request enabled.</td>
<td>CC3DE: Capture/Compare 3 DMA request enable</td>
<td>0: CC3 DMA request disabled.</td>
<td>1: CC3 DMA request enabled.</td>
<td>CC2DE: Capture/Compare 2 DMA request enable</td>
<td>0: CC2 DMA request disabled.</td>
<td>1: CC2 DMA request enabled.</td>
<td>CC1DE: Capture/Compare 1 DMA request enable</td>
<td>0: CC1 DMA request disabled.</td>
</tr>
</tbody>
</table>
43.4.5 TIMx status register (TIMx_SR)(x = 2 to 5)

Address offset: 0x10

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC4OF</td>
<td>CC3OF</td>
<td>CC2OF</td>
<td>CC1OF</td>
<td>TIF</td>
<td>CC4IF</td>
<td>CC3IF</td>
<td>CC2IF</td>
<td>CC1IF</td>
<td>UIF</td>
<td>CC4OF</td>
<td>CC3OF</td>
<td>CC2OF</td>
<td>CC1OF</td>
<td>UIF</td>
<td>UIF</td>
</tr>
<tr>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:13 Reserved, must be kept at reset value.

- Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
  - Refer to CC1OF description

- Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
  - Refer to CC1OF description

- Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
  - Refer to CC1OF description

- Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
  - This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
  - 0: No overcapture has been detected.
  - 1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.
Bit 6  **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5  Reserved, must be kept at reset value.

Bit 4  **CC4IF**: Capture/Compare 4 interrupt flag

Refer to CC1IF description

Bit 3  **CC3IF**: Capture/Compare 3 interrupt flag

Refer to CC1IF description

Bit 2  **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1  **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred
1: A compare match or an input capture occurred

**If channel CC1 is configured as output**: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

**If channel CC1 is configured as input**: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0  **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
   At overflow or underflow (for TIM2 to TIM4) and if UDIS=0 in the TIMx_CR1 register.
   When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
   When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

### 43.4.6  TIMx event generation register (TIMx_EGR)(x = 2 to 5)

Address offset: 0x14

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TG</td>
<td>CC4G</td>
<td>CC3G</td>
<td>CC2G</td>
<td>CC1G</td>
<td>UG</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>
Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
   This bit is set by software in order to generate an event, it is automatically cleared by hardware.
   0: No action
   1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
   Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
   Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
   Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
   This bit is set by software in order to generate an event, it is automatically cleared by hardware.
   0: No action
   1: A capture/compare event is generated on channel 1:
      **If channel CC1 is configured as output:**
      The CC1IF flag is set. Corresponding interrupt or DMA request is sent if enabled.
      **If channel CC1 is configured as input:**
      The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
   This bit can be set by software, it is automatically cleared by hardware.
   0: No action
   1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).
43.4.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 5)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output.
01: CC2 channel is configured as input, IC2 is mapped on TI2.
10: CC2 channel is configured as input, IC2 is mapped on TI1.
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).*
Bits 7:4  **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample T11 input and the length of the digital filter applied to T11. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- **0000**: No filter, sampling is done at fDTS
- **0001**: fSAMPLING=fCK_INT, N=2
- **0010**: fSAMPLING=fCK_INT, N=4
- **0011**: fSAMPLING=fCK_INT, N=8
- **0100**: fSAMPLING=fDTS/2, N=6
- **0101**: fSAMPLING=fDTS/2, N=8
- **0110**: fSAMPLING=fDTS/4, N=6
- **0111**: fSAMPLING=fDTS/4, N=8
- **1000**: fSAMPLING=fDTS/8, N=6
- **1001**: fSAMPLING=fDTS/8, N=8
- **1010**: fSAMPLING=fDTS/16, N=5
- **1011**: fSAMPLING=fDTS/16, N=6
- **1100**: fSAMPLING=fDTS/32, N=5
- **1101**: fSAMPLING=fDTS/32, N=6
- **1110**: fSAMPLING=fDTS/32, N=8

Bits 3:2  **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E=0 (TIMx_CCER register).

- **00**: no prescaler, capture is done each time an edge is detected on the capture input
- **01**: capture is done once every 2 events
- **10**: capture is done once every 4 events
- **11**: capture is done once every 8 events

Bits 1:0  **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- **00**: CC1 channel is configured as output
- **01**: CC1 channel is configured as input, IC1 is mapped on T11
- **10**: CC1 channel is configured as input, IC1 is mapped on T12
- **11**: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).*
43.4.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 5)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Output compare mode:**

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

Bits 31:25: Reserved, must be kept at reset value.

Bits 23:17: Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

**Note**: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCMR).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: OC1Ref is not affected by the ETRF input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input
Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT>TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

**Note**: In PWM mode, the OC1REF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.
**Bit 3** **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

**Bit 2** **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

**Bits 1:0** **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: CC1 channel is configured as input, IC1 is mapped on TI2.

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).*

### 43.4.9 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)

**x = 2 to 5**

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Input capture mode:**

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

- Bits 31:16: Reserved, must be kept at reset value.
- Bits 15:12: **IC4F[3:0]**: Input capture 4 filter
- Bits 11:10: **IC4PSC[1:0]**: Input capture 4 prescaler
43.4.10  **TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)**

(x = 2 to 5)

```plaintext
Address offset: 0x1C
Reset value: 0x0000 0000
```

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Output compare mode:**

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

- **Bits 31:25:** Reserved, must be kept at reset value.
- **Bits 23:17:** Reserved, must be kept at reset value.
- **Bit 15** **OC4CE:** Output compare 4 clear enable
- **Bits 24, 14:12** **OC4M[3:0]:** Output compare 4 mode
  Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)
- **Bit 11** **OC4PE:** Output compare 4 preload enable
- **Bit 10** **OC4FE:** Output compare 4 fast enable
43.4.11 **TIMx capture/compare enable register (TIMx_CCER)** \(x = 2\) to \(5\)

Address offset: 0x20

Reset value: 0x0000

<table>
<thead>
<tr>
<th>0000</th>
<th>0100</th>
<th>0110</th>
<th>1000</th>
<th>1010</th>
<th>1100</th>
<th>1110</th>
<th>1111</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC4NP</td>
<td>CC4P</td>
<td>CC4E</td>
<td>CC3NP</td>
<td>CC3P</td>
<td>CC3E</td>
<td>CC2NP</td>
<td>CC2P</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bit 15** **CC4NP**: Capture/Compare 4 output Polarity.
Refer to CC1NP description

**Bit 14** Reserved, must be kept at reset value.

**Bit 13** **CC4P**: Capture/Compare 4 output Polarity.
Refer to CC1P description

**Bit 12** **CC4E**: Capture/Compare 4 output enable.
refer to CC1E description

**Bit 11** **CC3NP**: Capture/Compare 3 output Polarity.
Refer to CC1NP description

**Bit 10** Reserved, must be kept at reset value.

**Bit 9** **CC3P**: Capture/Compare 3 output Polarity.
Refer to CC1P description

**Bit 8** **CC3E**: Capture/Compare 3 output enable.
Refer to CC1E description
General-purpose timers (TIM2/TIM3/TIM4/TIM5) RM0455

Bit 7 CC2NP: Capture/Compare 2 output Polarity.
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 CC2P: Capture/Compare 2 output Polarity.
refer to CC1P description

Bit 4 CC2E: Capture/Compare 2 output enable.
Refer to CC1E description

Bit 3 CC1NP: Capture/Compare 1 output Polarity.
CC1 channel configured as output: CC1NP must be kept cleared in this case.
CC1 channel configured as input: This bit is used in conjunction with CC1P to define
TI1FP1/TI2FP1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 CC1P: Capture/Compare 1 output Polarity.
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active
polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge
capture or trigger operations in reset, external clock or trigger mode),
TIxFP1 is not inverted (trigger operation in gated mode or encoder
mode).
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge
capture or trigger operations in reset, external clock or trigger mode),
TIxFP1 is inverted (trigger operation in gated mode or encoder
mode).
CC1NP=1, CC1P=0: non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising
and falling edges (capture or trigger operations in reset, external clock
or trigger mode), TIXFP1 is not inverted (trigger operation in gated
mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=1: This configuration is reserved, it must not be used.

Bit 0 CC1E: Capture/Compare 1 output enable.
0: Capture mode disabled / OC1 is not active
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

<table>
<thead>
<tr>
<th>CCxE bit</th>
<th>OCx output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output Disabled (OCx=0, OCx_EN=0)</td>
</tr>
<tr>
<td>1</td>
<td>OCx=OCxREF + Polarity, OCx_EN=1</td>
</tr>
</tbody>
</table>

Note: The state of the external IO pins connected to the standard OCx channels depends on the
OCx channel state and the GPIO and AFIO registers.

43.4.12 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 5)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in
TIMx_CR1 register:
• This section is for UIFREMAP = 0
• Next section is for UIFREMAP = 1
Address offset: 0x24
Reset value: 0x0000 0000

### 43.4.13 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 5)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx_CR1 register:
- Previous section is for UIFREMAP = 0
- This section is for UIFREMAP = 1

Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>CNT[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CNT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 **CNT[31:16]**: Most significant part counter value (TIM2 and TIM5)

Bits 15:0 **CNT[15:0]**: Least significant part of counter value

### 43.4.14 TIMx prescaler (TIMx_PSC)(x = 2 to 5)

Address offset: 0x28
Reset value: 0x0000

<table>
<thead>
<tr>
<th>PSC[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency $f_{\text{CK_CNT}}$ is equal to $f_{\text{CK_PSC}} / \left(\text{PSC}[15:0] + 1\right)$.
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in "reset mode").

43.4.15 TIMx auto-reload register (TIMx_ARR)(x = 2 to 5)
Address offset: 0x2C
Reset value: 0xFFFF FFFF

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

**ARR[31:16]**:

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>31:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 31:16</td>
<td><strong>ARR[31:16]</strong>: High auto-reload value (TIM2 and TIM5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 15 | 14 | 13 | 12 | 11 | 10 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1 |  0 |

**ARR[15:0]**:

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 31:16</td>
<td><strong>ARR[15:0]</strong>: Low Auto-reload value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ARR is the value to be loaded in the actual auto-reload register. Refer to the **Section 43.3.1: Time-base unit on page 1547** for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

43.4.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 5)
Address offset: 0x34
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>31:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 31:16</td>
<td><strong>CCR1[31:16]</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 31:16</td>
<td><strong>CCR1[15:0]</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:16  **CCR1[31:16]**: High Capture/Compare 1 value (TIM2 and TIM5)

Bits 15:0  **CCR1[15:0]**: Low Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

### 43.4.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 5)

Address offset: 0x38

Reset value: 0x0000 0000

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

**CCR2[31:16]**

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

Bits 31:16  **CCR2[31:16]**: High Capture/Compare 2 value (TIM2 and TIM5)

Bits 15:0  **CCR2[15:0]**: Low Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

### 43.4.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 5)

Address offset: 0x3C

Reset value: 0x0000 0000
General-purpose timers (TIM2/TIM3/TIM4/TIM5) RM0455

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

<table>
<thead>
<tr>
<th>CCR3[31:16]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>CCR4[31:16]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>CCR3[15:0]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
</table>

Bits 31:16 **CCR3[31:16]**: High Capture/Compare 3 value (TIM2 and TIM5)
Bits 15:0 **CCR3[15:0]**: Low Capture/Compare value

**If channel CC3 is configured as output:**
CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register
(bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when
an update event occurs.
The active capture/compare register contains the value to be compared to the counter
TIMx_CNT and signalled on OC3 output.

**If channel CC3 is configured as input:**
CCR3 is the counter value transferred by the last input capture 3 event (IC3). The
TIMx_CCR3 register is read-only and cannot be programmed.

43.4.19 **TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 5)**
Address offset: 0x40
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>CCR4[31:16]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>CCR4[15:0]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
</table>

Bits 31:16 **CCR4[31:16]**: High Capture/Compare 4 value (TIM2 and TIM5)
Bits 15:0 **CCR4[15:0]**: Low Capture/Compare value

1. **if CC4 channel is configured as output (CC4S bits):**
CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register
(bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when
an update event occurs.
The active capture/compare register contains the value to be compared to the counter
TIMx_CNT and signalled on OC4 output.

2. **if CC4 channel is configured as input (CC4S bits in TIMx_CCMR4 register):**
CCR4 is the counter value transferred by the last input capture 4 event (IC4). The
TIMx_CCR4 register is read-only and cannot be programmed.
43.4.20 TIMx DMA control register (TIMx_DCR)(x = 2 to 5)

Address offset: 0x48
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits 15:13</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 12:8</td>
<td><strong>DBL[4:0]</strong>: DMA burst length</td>
</tr>
<tr>
<td></td>
<td>This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).</td>
</tr>
<tr>
<td></td>
<td>00000: 1 transfer,</td>
</tr>
<tr>
<td></td>
<td>00001: 2 transfers,</td>
</tr>
<tr>
<td></td>
<td>00010: 3 transfers,</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>10001: 18 transfers.</td>
</tr>
<tr>
<td>Bits 7:5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 4:0</td>
<td><strong>DBA[4:0]</strong>: DMA base address</td>
</tr>
<tr>
<td></td>
<td>This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>00000: TIMx_CR1</td>
</tr>
<tr>
<td></td>
<td>00001: TIMx_CR2</td>
</tr>
<tr>
<td></td>
<td>00010: TIMx_SMCR</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Example: Let us consider the following transfer: DBL = 7 transfers &amp; DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.</td>
</tr>
</tbody>
</table>

43.4.21 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)

Address offset: 0x4C
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th><strong>DMAB[15:0]</strong>: DMA register for burst accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4 where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).</td>
</tr>
</tbody>
</table>
### 43.4.22 TIM2 alternate function option register 1 (TIM2_AF1)

Address offset: 0x60  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17:14</td>
<td><strong>ETRSEL[3:0]</strong> ETR source selection</td>
</tr>
<tr>
<td></td>
<td>These bits select the ETR input source.</td>
</tr>
<tr>
<td></td>
<td>0000: ETR input is connected to I/O</td>
</tr>
<tr>
<td></td>
<td>0001: COMP1 output</td>
</tr>
<tr>
<td></td>
<td>0010: COMP2 output</td>
</tr>
<tr>
<td></td>
<td>0011: LSE</td>
</tr>
<tr>
<td></td>
<td>0100: SAI1 FS_A</td>
</tr>
<tr>
<td></td>
<td>0101: SAI1 FS_B</td>
</tr>
<tr>
<td></td>
<td>Others: Reserved</td>
</tr>
<tr>
<td>13:0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

### 43.4.23 TIM3 alternate function option register 1 (TIM3_AF1)

Address offset: 0x60  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17:14</td>
<td><strong>ETRSEL[3:0]</strong> ETR source selection</td>
</tr>
<tr>
<td></td>
<td>These bits select the ETR input source.</td>
</tr>
<tr>
<td></td>
<td>0000: ETR input is connected to I/O</td>
</tr>
<tr>
<td></td>
<td>0001: COMP1 output</td>
</tr>
<tr>
<td></td>
<td>0010: COMP2 output</td>
</tr>
<tr>
<td></td>
<td>0011: LSE</td>
</tr>
<tr>
<td></td>
<td>0100: SAI1 FS_A</td>
</tr>
<tr>
<td></td>
<td>0101: SAI1 FS_B</td>
</tr>
<tr>
<td></td>
<td>Others: Reserved</td>
</tr>
<tr>
<td>13:0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

### 43.4.24 TIM4 alternate function option register 1 (TIM4_AF1)

Address offset: 0x60  
Reset value: 0x0000 0000
43.4.25 TIM5 alternate function option register 1 (TIM5_AF1)

Address offset: 0x60
Reset value: 0x0000 0000

| Bits 31:18 | Reserved, must be kept at reset value. |
| Bits 17:14 | **ETRSEL[3:0]** ETR source selection |
| | These bits select the ETR input source. |
| | 0000: ETR input is connected to I/O |
| | Others: Reserved |
| Bits 13:0 | Reserved, must be kept at reset value. |

43.4.26 TIM2 timer input selection register (TIM2_TISEL)

Address offset: 0x68
Reset value: 0x0000 0000

| Bits 31:18 | Reserved, must be kept at reset value. |
| Bits 17:14 | **ETRSEL[3:0]** ETR source selection |
| | These bits select the ETR input source. |
| | 0000: ETR input is connected to I/O |
| | 0001: SAI2 FS_A connected to ETR input |
| | 0010: SAI2 FS_B connected to ETR input |
| | Others: Reserved |
| Bits 13:0 | Reserved, must be kept at reset value. |
43.4.27 TIM3 timer input selection register (TIM3_TISEL)

Address offset: 0x68
Reset value: 0x0000 0000

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

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

Bits 27:24 **TIM3SEL[3:0]**: Ti4[0] to Ti4[15] input selection
These bits select the Ti4[0] to Ti4[15] input source.
0000: TIM2_CH4 input
0001: COMP1 output
0010: COMP2 output
0011: COMP1 output OR COMP2 output
Others: Reserved

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

Bits 19:16 **TIM3SEL[3:0]**: Ti3[0] to Ti3[15] input selection
These bits select the Ti3[0] to Ti3[15] input source.
0000: TIM2_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TIM3SEL[3:0]**: Ti2[0] to Ti2[15] input selection
These bits select the Ti2[0] to Ti2[15] input source.
0000: TIM2_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TIM3SEL[3:0]**: Ti1[0] to Ti1[15] input selection
These bits select the Ti1[0] to Ti1[15] input source.
0000: TIM2_CH1 input
Others: Reserved

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

Bits 27:24 **TIM4SEL[3:0]**: Ti4[0] to Ti4[15] input selection
These bits select the Ti4[0] to Ti4[15] input source.
0000: TIM3_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.
43.4.28  **TIM4 timer input selection register (TIM4_TISEL)**

Address offset: 0x68

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

Bits 19:16  **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection
These bits select the TI3[0] to TI3[15] input source.
0000: TIM3_CH3 input
Others: Reserved

Bits 15:12  Reserved, must be kept at reset value.

Bits 11:8  **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection
These bits select the TI2[0] to TI2[15] input source.
0000: TIM3_CH2 input
Others: Reserved

Bits 7:4  Reserved, must be kept at reset value.

Bits 3:0  **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection
These bits select the TI1[0] to TI1[15] input source.
0000: TIM3_CH1 input
0001: COMP1 output
0010: COMP2 output
0011: COMP1 output OR COMP2 output
Others: Reserved

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

Bits 27:24  **TI4SEL[3:0]**: TI4[0] to TI4[15] input selection
These bits select the TI4[0] to TI4[15] input source.
0000: TIM4_CH4 input
Others: Reserved

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

Bits 19:16  **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection
These bits select the TI3[0] to TI3[15] input source.
0000: TIM4_CH3 input
Others: Reserved

Bits 15:12  Reserved, must be kept at reset value.
### 43.4.29 TIM5 timer input selection register (TIM5_TISEL)

**Address offset:** 0x68

**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Default Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-28</td>
<td>Reserved</td>
<td>0x0000 0000</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>27-24</td>
<td><strong>TI4SEL[3:0]</strong></td>
<td>0x0000</td>
<td><strong>TIM5_CH4</strong> input. Others: Reserved.</td>
</tr>
<tr>
<td>23-20</td>
<td>Reserved</td>
<td>0x0000</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>19-16</td>
<td><strong>TI3SEL[3:0]</strong></td>
<td>0x0000</td>
<td><strong>TIM5_CH3</strong> input. Others: Reserved.</td>
</tr>
<tr>
<td>15-12</td>
<td>Reserved</td>
<td>0x0000</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>11-8</td>
<td><strong>TI2SEL[3:0]</strong></td>
<td>0x0000</td>
<td><strong>TIM5_CH2</strong> input. Others: Reserved.</td>
</tr>
<tr>
<td>7-4</td>
<td>Reserved</td>
<td>0x0000</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>3-0</td>
<td><strong>TI1SEL[3:0]</strong></td>
<td>0x0000</td>
<td><strong>fdcan1_tmp</strong>. <strong>fdcan1_rtp</strong>. Others: Reserved.</td>
</tr>
</tbody>
</table>
### 43.4.30 TIMx register map

TIMx registers are mapped as described in the table below:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>TIMx_CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>TIMx_CR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>TIMx_SMCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>TIMx_DIER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>TIMx_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>TIMx_EGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>TIMx_CCMR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>TIMx_CCMR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>TIMx_CCER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 312. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>CNT[30:16]</th>
<th>CNT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x24</td>
<td>TIMx_CNT</td>
<td>TIMx_CNT</td>
<td>CNT[30:16]</td>
<td>CNT[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIMx_CNT</td>
<td>(TIM2 and TIM5 only, reserved on the other timers)</td>
<td>(TIM2 and TIM5 only, reserved on the other timers)</td>
</tr>
<tr>
<td>0x28</td>
<td>TIMx_PSC</td>
<td>TIMx_PSC</td>
<td>PSC[15:0]</td>
<td></td>
</tr>
<tr>
<td>0x2C</td>
<td>TIMx_ARR</td>
<td>TIMx_ARR</td>
<td>ARR[31:16]</td>
<td>ARR[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIMx_ARR</td>
<td>(TIM2 and TIM5 only, reserved on the other timers)</td>
<td>(TIM2 and TIM5 only, reserved on the other timers)</td>
</tr>
<tr>
<td>0x30</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x34</td>
<td>TIMx_CCR1</td>
<td>TIMx_CCR1</td>
<td>CCR1[31:16]</td>
<td>CCR1[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIMx_CCR1</td>
<td>(TIM2 and TIM5 only, reserved on the other timers)</td>
<td>(TIM2 and TIM5 only, reserved on the other timers)</td>
</tr>
<tr>
<td>0x38</td>
<td>TIMx_CCR2</td>
<td>TIMx_CCR2</td>
<td>CCR2[31:16]</td>
<td>CCR2[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIMx_CCR2</td>
<td>(TIM2 and TIM5 only, reserved on the other timers)</td>
<td>(TIM2 and TIM5 only, reserved on the other timers)</td>
</tr>
<tr>
<td>0x3C</td>
<td>TIMx_CCR3</td>
<td>TIMx_CCR3</td>
<td>CCR3[31:16]</td>
<td>CCR3[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIMx_CCR3</td>
<td>(TIM2 and TIM5 only, reserved on the other timers)</td>
<td>(TIM2 and TIM5 only, reserved on the other timers)</td>
</tr>
<tr>
<td>0x40</td>
<td>TIMx_CCR4</td>
<td>TIMx_CCR4</td>
<td>CCR4[31:16]</td>
<td>CCR4[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIMx_CCR4</td>
<td>(TIM2 and TIM5 only, reserved on the other timers)</td>
<td>(TIM2 and TIM5 only, reserved on the other timers)</td>
</tr>
<tr>
<td>0x44</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x48</td>
<td>TIMx_DCR</td>
<td>TIMx_DCR</td>
<td>DBL[4:0]</td>
<td>DBA[4:0]</td>
</tr>
<tr>
<td>0x4C</td>
<td>TIMx_DMAR</td>
<td>TIMx_DMAR</td>
<td>DMAB[15:0]</td>
<td></td>
</tr>
<tr>
<td>0x60</td>
<td>TIM2_AF1</td>
<td>TIM2_AF1</td>
<td>ETRSEL[3:0]</td>
<td></td>
</tr>
<tr>
<td>0x60</td>
<td>TIM3_AF1</td>
<td>TIM3_AF1</td>
<td>ETRSEL[3:0]</td>
<td></td>
</tr>
</tbody>
</table>
### Table 312. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x60   | TIM4_AF1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x60   | TIM5_AF1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x68   | TIM2_TISEL    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI4SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI3SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI2SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI1SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x68   | TIM3_TISEL    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI4SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI3SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI2SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI1SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x68   | TIM4_TISEL    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI4SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI3SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI2SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI1SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x68   | TIM5_TISEL    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI4SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI3SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI2SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TI1SEL[3:0]   | 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.3 on page 128 for the register boundary addresses.
44 General-purpose timers (TIM12/TIM13/TIM14)

44.1 TIM12/TIM13/TIM14 introduction

The TIM12/TIM13/TIM14 general-purpose timers consist in a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM12/TIM13/TIM14 timers are completely independent, and do not share any resources. They can be synchronized together as described in Section 44.3.17: Timer synchronization (TIM12).

44.2 TIM12/TIM13/TIM14 main features

44.2.1 TIM12 main features

The features of the TIM12 general-purpose timer include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization or count by internal trigger)
  - Input capture
  - Output compare
### 44.2.2 TIM13/TIM14 main features

The features of general-purpose timers TIM13/TIM14 include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare
Figure 423. General-purpose timer block diagram (TIM13/TIM14)

Notes:
- Preload registers transferred to active registers on U event according to control bit
- Event
- Interrupt & DMA output
44.3 TIM12/TIM13/TIM14 functional description

44.3.1 Time-base unit

The main block of the timer is a 16-bit up-counter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:
- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 424* and *Figure 425* give some examples of the counter behavior when the prescaler ratio is changed on the fly.
**Figure 424. Counter timing diagram with prescaler division change from 1 to 2**

- CK_PSC
- CEN
- Timer clock = CK_CNT
- Counter register
  - F7 F8 F9 FA FB FC 00 01 02 03
- Update event (UEV)
- Prescaler control register
  - Write a new value in TIMx_PSC
  - 0 1
- Prescaler buffer
  - 0 1
- Prescaler counter
  - 0 0 1 0 1 0 1

**Figure 425. Counter timing diagram with prescaler division change from 1 to 4**

- CK_PSC
- CEN
- Timer clock = CK_CNT
- Counter register
  - F7 F8 F9 FA FB FC 00 01
- Update event (UEV)
- Prescaler control register
  - Write a new value in TIMx_PSC
  - 0 3
- Prescaler buffer
  - 0 3
- Prescaler counter
  - 0 0 1 2 3 0 1 2 3
44.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller on TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 426. Counter timing diagram, internal clock divided by 1

![Counter timing diagram](image-url)
Figure 427. Counter timing diagram, internal clock divided by 2

Figure 428. Counter timing diagram, internal clock divided by 4
**Figure 429. Counter timing diagram, internal clock divided by N**

- **CK_PSC**
- **Timer clock = CK_CNT**
- **Counter register**
- **Counter overflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**

**Figure 430. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)**

- **CK_PSC**
- **CEN**
- **Timer clock = CK_CNT**
- **Counter register**
- **Counter overflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**
- **Auto-reload preload register**

Write a new value in TIMx_ARR
44.3.3 Clock selection

The counter clock can be provided by the following clock sources:
- Internal clock (CK_INT)
- External clock mode1 (for TIM12): external input pin (TIx)
- Internal trigger inputs (ITRx) (for TIM12): connecting the trigger output from another timer. For instance, another timer can be configured as a prescaler for TIM12. Refer to Section: Using one timer as prescaler for another timer for more details.

Internal clock source (CK_INT)

The internal clock source is the default clock source for TIM13/TIM14.

For TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS='000'). The CEN bit in the TIMx_CR1 register and the UG bit in the TIMx_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock CK_INT.

*Figure 432* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.
External clock source mode 1 (TIM12)

This mode is selected when SMS='111' in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 433. TI2 external clock connection example

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:
1. Select the proper TII[x] source (internal or external) with the TII2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TII2 input by writing CC2S = '01' in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F='0000').
4. Select the rising edge polarity by writing CC2P='0' and CC2NP='0' in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS='111' in the TIMx_SMCR register.
6. Select TII2 as the trigger input source by writing TS='110' in the TIMx_SMCR register.
7. Enable the counter by writing CEN='1' in the TIMx_CR1 register.

**Note:** *The capture prescaler is not used for triggering, so it does not need to be configured.*

When a rising edge occurs on TII2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TII2 and the actual clock of the counter is due to the resynchronization circuit on TII2 input.

### Figure 434. Control circuit in external clock mode 1

<table>
<thead>
<tr>
<th>TII2</th>
<th>CNT_EN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Counter clock = CK_CNT = CK_PSC

Counter register

<table>
<thead>
<tr>
<th>34</th>
<th>35</th>
<th>36</th>
</tr>
</thead>
</table>

Write TIF=0

#### 44.3.4 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

*Figure 435 to Figure 437* give an overview of one capture/compare channel.

The input stage samples the corresponding TIIx input to generate a filtered signal TIIxF. Then, an edge detector with polarity selection generates a signal (TIIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).
The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 436. Capture/compare channel 1 main circuit
44.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1[x] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to ‘01’ in the TIMx_CCMR1 register. As soon as CC1S becomes different from ‘00’, the channel is configured in input mode and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (by programming the ICxF bits in the TIMx_CCMR1 register if the input is one of the TIx inputs). Let’s imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the
new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to
‘0011’ in the TIMx_CCMR1 register.

4. Select the edge of the active transition on the TI1 channel by programming CC1P and
CC1NP bits to ‘00’ in the TIMx_CCER register (rising edge in this case).

5. Program the input prescaler. In our example, we wish the capture to be performed at
each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the
TIMx_CCMR1 register).

6. Enable capture from the counter into the capture register by setting the CC1E bit in the
TIMx_CCER register.

7. If needed, enable the related interrupt request by setting the CC1IE bit in the
TIMx_DIER register.

When an input capture occurs:
- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures
  occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the
overcapture flag. This is to avoid missing an overcapture which could happen after reading
the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in
the TIMx_EGR register.

44.3.6 PWM input mode (only for TIM12)

This mode is a particular case of input capture mode. The procedure is the same except:
- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller
  is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in
TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending
on CK_INT frequency and prescaler value):
1. Select the proper TI1[x] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to ‘01’ in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): program the CC1P and CC1NP bits to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to ‘10’ in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): program the CC2P and CC2NP bits to ‘1’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to ‘00101’ in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to ‘100’ in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

**Figure 438. PWM input mode timing**

1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

### 44.3.7 Forced output mode

In output mode (CCxS bits = ‘00’ in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCxREF/OCx) to its active level, one just needs to write ‘0101’ in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP='0' (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to ‘0100’ in the TIMx_CCMRx register.
Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

### 44.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM='0000'), be set active (OCxM='0001'), be set inactive (OCxM='0010') or can toggle (OCxM='0011') on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = '0011' to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = '0' to disable preload register
   - Write CCxP = '0' to select active high polarity
   - Write CCxE = '1' to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 439.
44.3.9 PWM mode

Pulse Width Modulation mode allows to generate a signal with a frequency determined by the value of the TIMx.ARR register and a duty cycle determined by the value of the TIMx.CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx.CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx.CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx.CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx.EGR register.

The OCx polarity is software programmable using the CCxP bit in the TIMx.CCER register. It can be programmed as active high or active low. The OCx output is enabled by the CCxE bit in the TIMx.CCER register. Refer to the TIMx.CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx.CCRx are always compared to determine whether TIMx_CNT ≤ TIMx.CCRx.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx.CCRx else it becomes low. If the compare value in TIMx.CCRx is greater than the auto-reload value (in TIMx.ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. Figure 440 shows some edge-aligned PWM waveforms in an example where TIMx.ARR=8.
44.3.10 Combined PWM mode (TIM12 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1100' (Combined PWM mode 1) or '1101' (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

**Note:** The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

*Figure 441* represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
44.3.11 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

\[ \text{CNT} < \text{CCRx} \leq \text{ARR} \] (in particular, \( 0 < \text{CCRx} \))
For example one may want to generate a positive pulse on OC1 with a length of $t_{\text{PULSE}}$ and after a delay of $t_{\text{DELAY}}$ as soon as a positive edge is detected on the TI2 input pin.

Use TI2FP2 as trigger 1:
1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='00110' in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to ‘110’ in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).
- The $t_{\text{DELAY}}$ is defined by the value written in the TIMx_CCR1 register.
- The $t_{\text{PULSE}}$ is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let’s say one want to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M='0111' in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0’ in this example.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to ‘0’, so the Repetitive Mode is selected.
Particular case: OCx fast enable
In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay $t_{\text{DELAY} \text{ min}}$ we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

44.3.12 Retriggerable one pulse mode (OPM) (TIM12 only)
This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one pulse mode described in Section 44.3.11: One-pulse mode:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for retrigerrable OPM mode 1 or 2.

If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.
44.3.13 **UIF bit remapping**

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

44.3.14 **Timer input XOR function**

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins TIMx_CH1 and TIMx_CH2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in Figure 444.

*Figure 444. Measuring time interval between edges on 2 signals*

44.3.15 **TIM12 external trigger synchronization**

The TIM12 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

**Slave mode: Reset mode**

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=’0000’). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = ’01’ in the TIMx_CCRM1 register.
Program CC1P and CC1NP to ‘00’ in TIMx_CCER register to validate the polarity (and detect rising edges only).

2. Configure the timer in reset mode by writing SMS='100' in TIMx_SMCR register. Select TI1 as the input source by writing TS='00101' in TIMx_SMCR register.

3. Start the counter by writing CEN='1' in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

![Figure 445. Control circuit in reset mode](image)

**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F='0000'). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S='01' in TIMx_CCMR1 register. Program CC1P='1' and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS='101' in TIMx_SMCR register. Select TI1 as the input source by writing TS='00101' in TIMx_SMCR register.

3. Enable the counter by writing CEN='1' in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN='0', whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.
The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

**Figure 446. Control circuit in gated mode**

```
<table>
<thead>
<tr>
<th>T11</th>
<th>cnt_en</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Counter clock = ck_cnt = ck_psc

Counter register

<table>
<thead>
<tr>
<th>30</th>
<th>31</th>
<th>32</th>
<th>33</th>
<th>34</th>
<th>35</th>
<th>36</th>
<th>37</th>
<th>38</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TIF

Write TIF=0
```

**Slave mode: Trigger mode**

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=’0000’). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=’01’ in TIMx_CCMR1 register. Program CC2P=’1’ and CC2NP=’0’ in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS=’110’ in TIMx_SMCR register. Select TI2 as the input source by writing TS=’00110’ in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.
44.3.16 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.
44.3.17 Timer synchronization (TIM12)

The TIM timers are linked together internally for timer synchronization or chaining. Refer to Section 43.3.19: Timer synchronization for details.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

44.3.18 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBGMCU module. For more details, refer to Section 64.5.7: Microcontroller debug unit (DBGMCU).

44.4 TIM12 registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

44.4.1 TIM12 control register 1 (TIM12_CR1)

Address offset: 0x00

Reset value: 0x0000

| Bits 15:12 | Reserved, must be kept at reset value. |
| Bits 11  | UIFREMAP: UIF status bit remapping |
| 0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31. |
| 1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31. |
| Bits 10  | Reserved, must be kept at reset value. |
| Bits 9:8 | CKD[1:0]: Clock division |
| 00: tDTS = tCK_INT |
| 01: tDTS = 2 x tCK_INT |
| 10: tDTS = 4 x tCK_INT |
| 11: Reserved |
| Bits 7  | ARPE: Auto-reload preload enable |
| 0: TIMx_ARR register is not buffered. |
| 1: TIMx_ARR register is buffered. |
| Bits 6:4 | Reserved, must be kept at reset value. |
44.4.2 TIM12 control register 2 (TIM12_CR2)

Address offset: 0x04
Reset value: 0x0000

Bit 3 **OPM**: One-pulse mode
   - 0: Counter is not stopped on the update event
   - 1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source
   - This bit is set and cleared by software to select the UEV event sources.
   - 0: Any of the following events generates an update interrupt if enabled. These events can be:
     - Counter overflow
     - Setting the UG bit
     - Update generation through the slave mode controller
   - 1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable
   - This bit is set and cleared by software to enable/disable update event (UEV) generation.
   - 0: UEV enabled. An UEV is generated by one of the following events:
     - Counter overflow
     - Setting the UG bit
     - Buffered registers are then loaded with their preload values.
   - 1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable
   - 0: Counter disabled
   - 1: Counter enabled
   - CEN is cleared automatically in one-pulse mode, when an update event occurs.

Note: *External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*
44.4.3 TIM12 slave mode control register (TIM12_SMCR)

Address offset: 0x08
Reset value: 0x0000 0000

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

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

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM**: Master/Slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful in order to synchronize several timers on a single external event.
Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This TS[4:0] bitfield selects the trigger input to be used to synchronize the counter.

- 00000: Internal Trigger 0 (ITR0)
- 00001: Internal Trigger 1 (ITR1)
- 00010: Internal Trigger 2 (ITR2)
- 00011: Internal Trigger 3 (ITR3)
- 00100: TI1 Edge Detector (TI1F_ED)
- 00101: Filtered Timer Input 1 (TI1FP1)
- 00110: Filtered Timer Input 2 (TI2FP2)
- Others: Reserved

See Table 313: TIMx internal trigger connection on page 1648 for more details on the meaning of ITRx for each timer.

**Note:** These bits must be changed only when they are not used (e.g. when SMS='000') to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

- 0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.
- 0001: Reserved
- 0010: Reserved
- 0011: Reserved
- 0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
- 0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
- 0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
- 0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
- 1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.
- Other codes: reserved.

**Note:** The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=’00100’). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

**Note:** The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

### Table 313. TIMx internal trigger connection

<table>
<thead>
<tr>
<th>Slave TIM</th>
<th>ITR0 (TS = ‘00000’)</th>
<th>ITR1 (TS = ‘00001’)</th>
<th>ITR2 (TS = ‘00010’)</th>
<th>ITR3 (TS = ‘00011’)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM12</td>
<td>TIM4</td>
<td>TIM5</td>
<td>TIM13 OC1</td>
<td>TIM14 OC1</td>
</tr>
</tbody>
</table>

1648/2974  RM0455 Rev 3
### 44.4.4 TIM12 Interrupt enable register (TIM12_DIER)

Address offset: 0x0C  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIE</td>
<td>CC2IE</td>
<td>CC1IE</td>
<td>UIE</td>
<td>CC2OF</td>
<td>CC1OF</td>
<td>TIF</td>
<td>CC2IF</td>
<td>CC1IF</td>
<td>UIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Bits 15:7** Reserved, must be kept at reset value.
- **Bit 6** TIE: Trigger interrupt enable
  - 0: Trigger interrupt disabled.
  - 1: Trigger interrupt enabled.
- **Bits 5:3** Reserved, must be kept at reset value.
- **Bit 2** CC2IE: Capture/Compare 2 interrupt enable
  - 0: CC2 interrupt disabled.
  - 1: CC2 interrupt enabled.
- **Bit 1** CC1IE: Capture/Compare 1 interrupt enable
  - 0: CC1 interrupt disabled.
  - 1: CC1 interrupt enabled.
- **Bit 0** UIE: Update interrupt enable
  - 0: Update interrupt disabled.
  - 1: Update interrupt enabled.

### 44.4.5 TIM12 status register (TIM12_SR)

Address offset: 0x10  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC2OF</td>
<td>CC1OF</td>
<td>TIF</td>
<td>CC2IF</td>
<td>CC1IF</td>
<td>UIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Bits 15:11** Reserved, must be kept at reset value.
- **Bit 10** CC2OF: Capture/compare 2 overcapture flag
  - refer to CC1OF description
- **Bit 9** CC1OF: Capture/Compare 1 overcapture flag
  - This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
  - 0: No overcapture has been detected.
  - 1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set
- **Bits 8:7** Reserved, must be kept at reset value.
44.4.6 TIM12 event generation register (TIM12_EGR)

Address offset: 0x14
Reset value: 0x0000

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

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

Bit 6 TG: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in the TIMx_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.
Bit 2 **CC2G**: Capture/compare 2 generation  
refer to CC1G description  

Bit 1 **CC1G**: Capture/compare 1 generation  
This bit is set by software to generate an event, it is automatically cleared by hardware.  
0: No action  
1: A capture/compare event is generated on channel 1:  
  **If channel CC1 is configured as output:**  
  the CC1IF flag is set, the corresponding interrupt is sent if enabled.  
  **If channel CC1 is configured as input:**  
  The current counter value is captured in the TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.  

Bit 0 **UG**: Update generation  
This bit can be set by software, it is automatically cleared by hardware.  
0: No action  
1: Re-initializes the counter and generates an update of the registers. The prescaler counter is also cleared and the prescaler ratio is not affected. The counter is cleared.  

**44.4.7 TIM12 capture/compare mode register 1 [alternate]**  
**(TIM12_CCMR1)**  
Address offset: 0x18  
Reset value: 0x0000 0000  
The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.  

**Input capture mode:**  

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

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

Bits 31:16: Reserved, must be kept at reset value.  

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter  

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler  

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection  
This bitfield defines the direction of the channel (input/output) as well as the used input.  
00: CC2 channel is configured as output  
01: CC2 channel is configured as input, IC2 is mapped on TI2  
10: CC2 channel is configured as input, IC2 is mapped on TI1  
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)  

*Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).*
44.4.8 TIM12 capture/compare mode register 1 [alternate]
(TIM12_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the

Bits 7:4  **IC1F[3:0]:** Input capture 1 filter

This bitfield defines the frequency used to sample the TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at fDTS
- 0001: fSAMPLING=fCK_INT, N=2
- 0010: fSAMPLING=fCK_INT, N=4
- 0011: fSAMPLING=fCK_INT, N=8
- 0100: fSAMPLING=fDTS/2, N=6
- 0101: fSAMPLING=fDTS/2, N=8
- 0110: fSAMPLING=fDTS/4, N=6
- 0111: fSAMPLING=fDTS/4, N=8
- 1000: fSAMPLING=fDTS/8, N=6
- 1001: fSAMPLING=fDTS/8, N=8
- 1010: fSAMPLING=fDTS/16, N=5
- 1011: fSAMPLING=fDTS/16, N=6
- 1100: fSAMPLING=fDTS/16, N=8
- 1101: fSAMPLING=fDTS/32, N=5
- 1110: fSAMPLING=fDTS/32, N=6
- 1111: fSAMPLING=fDTS/32, N=8

Bits 3:2  **IC1PSC[1:0]:** Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).

The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0  **CC1S[1:0]:** Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, IC1 is mapped on TI1
- 10: CC1 channel is configured as input, IC1 is mapped on TI2
- 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).
corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

**Output compare mode:**

<table>
<thead>
<tr>
<th></th>
<th>OC2M[3:0]</th>
<th>OC2PE</th>
<th>OC2FE</th>
<th>CC2S[1:0]</th>
<th>OC1M[2:0]</th>
<th>OC1PE</th>
<th>OC1FE</th>
<th>CC1S[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

Refer to OC1M[3:0] for bit description.

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

*Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).*

Bit 7 Reserved, must be kept at reset value.
Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas the active level of OC1 depends on the CC1P.

- **0000**: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. (this mode is used to generate a timing base).
- **0001**: Set channel 1 to active level on match. The OC1REF signal is forced high when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).
- **0010**: Set channel 1 to inactive level on match. The OC1REF signal is forced low when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).
- **0011**: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1
- **0100**: Force inactive level - OC1REF is forced low
- **0101**: Force active level - OC1REF is forced high
- **0110**: PWM mode 1 - channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else it is inactive
- **0111**: PWM mode 2 - channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else it is active
- **1000**: Retriggable OPM mode 1 - The channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.
- **1001**: Retriggable OPM mode 2 - The channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update.
- **1010**: Reserved,
- **1011**: Reserved,
- **1100**: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.
- **1101**: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.
- **1110**: Reserved,
- **1111**: Reserved

**Note:** In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.
Bit 3 **OC1PE**: Output compare 1 preload enable
- 0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken into account immediately
- 1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded into the active register at each update event

*Note: The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in the TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output compare 1 fast enable
- This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
- 0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the CC1 output when an edge occurs on the trigger input is 5 clock cycles
- 1: An active edge on the trigger input acts like a compare match on the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection
- This bitfield defines the direction of the channel (input/output) as well as the used input.
  - 00: CC1 channel is configured as output
  - 01: CC1 channel is configured as input, IC1 is mapped on TI1
  - 10: CC1 channel is configured as input, IC1 is mapped on TI2
  - 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

*Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).*

### 44.4.9 TIM12 capture/compare enable register (TIM12_CCER)

Address offset: 0x20
Reset value: 0x0000

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

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity
- Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity
- Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
- Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity
- CC1 channel configured as output: CC1NP must be kept cleared
- CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity (refer to CC1P description).
Bit 2 Reserved, must be kept at reset value.

Bit 1 CC1P: Capture/Compare 1 output Polarity.
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode). TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode). TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
CC1NP=1, CC1P=0: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode). TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=1: This configuration is reserved, it must not be used.

Bit 0 CC1E: Capture/Compare 1 output enable.
0: Capture mode disabled / OC1 is not active
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 314. Output control bit for standard OCx channels

<table>
<thead>
<tr>
<th>CCxE bit</th>
<th>OCx output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output disabled (OCx='0', OCx_EN='0')</td>
</tr>
<tr>
<td>1</td>
<td>OCx=OCxREF + Polarity, OCx_EN='1'</td>
</tr>
</tbody>
</table>

Note: The states of the external I/O pins connected to the standard OCx channels depend on the state of the OCx channel and on the GPIO registers.

44.4.10 TIM12 counter (TIM12_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30:16</th>
<th>Bit 15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>UIFCY</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31 UIFCY: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.
Bits 30:16 Reserved, must be kept at reset value.
Bits 15:0 CNT[15:0]: Counter value
44.4.11 TIM12 prescaler (TIM12_PSC)
Address offset: 0x28
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15-0</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSC[15-0]</td>
<td>Prescaler value</td>
<td>The counter clock frequency CK_CNT is equal to fCK_PSC / (PSC[15-0] + 1). PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in &quot;reset mode&quot;).</td>
</tr>
</tbody>
</table>

44.4.12 TIM12 auto-reload register (TIM12_ARR)
Address offset: 0x2C
Reset value: 0xFFFF

<table>
<thead>
<tr>
<th>Bit 15-0</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARR[15-0]</td>
<td>Auto-reload value</td>
<td>ARR is the value to be loaded into the actual auto-reload register. Refer to the Section 44.3.1: Time-base unit on page 1623 for more details about ARR update and behavior. The counter is blocked while the auto-reload value is null.</td>
</tr>
</tbody>
</table>

44.4.13 TIM12 capture/compare register 1 (TIM12_CCR1)
Address offset: 0x34
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15-0</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCR1[15-0]</td>
<td>Capture/Compare 1 value</td>
<td>If channel CC1 is configured as output: CCR1 is the value to be loaded into the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (OC1PE bit). Else the preload value is copied into the active capture/compare 1 register when an update event occurs. The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signaled on the OC1 output. If channel CC1 is configured as input: CCR1 is the counter value transferred by the last input capture 1 event (IC1).</td>
</tr>
</tbody>
</table>
44.4.14 TIM12 capture/compare register 2 (TIM12_CCR2)

Address offset: 0x38
Reset value: 0x0000

<table>
<thead>
<tr>
<th>CCR2[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:0 **CCR2[15:0]: Capture/Compare 2 value**

*If channel CC2 is configured as output:*
CCR2 is the value to be loaded into the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (OC2PE bit). Else the preload value is copied into the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signalled on the OC2 output.

*If channel CC2 is configured as input:*
CCR2 is the counter value transferred by the last input capture 2 event (IC2).

44.4.15 TIM12 timer input selection register (TIM12_TISEL)

Address offset: 0x68
Reset value: 0x0000

<table>
<thead>
<tr>
<th>TI2SEL[3:0]</th>
<th>TI1SEL[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:12 **Reserved, must be kept at reset value.**

Bits 11:8 **TI2SEL[3:0]:** selects TI2[0] to TI2[15] input
0000: TIM12_CH2 input
Other: Reserved

Bits 7:4 **Reserved, must be kept at reset value.**

Bits 3:0 **TI1SEL[3:0]:** selects TI1[0] to TI1[15] input
0000: TIM12_CH1 input
0001: spdifrx_frame_sync
Other: Reserved
### 44.4.16 TIM12 register map

TIM12 registers are mapped as 16-bit addressable registers as described below:

| Offset | Register name  | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x00   | TIMx_CR1       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x04   | TIM12_CR2      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x08   | TIMx_SMCR      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0C   | TIMx_DIER      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x10   | TIMx_SR        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x14   | TIMx_EGR       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x18   | TIMx_CCMR1     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Output Compare |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | mode           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1C   | TIMx_CCR1      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reserved       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x20   | TIMx_CCR2      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x24   | TIMx_CNT       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | UF/CPY         |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x28   | TIMx_PSC       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | PSC[15:0]      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

**Table 315. TIM12 register map and reset values**
Table 315. TIM12 register map and reset values (continued)

| Offset | Register name | Offset name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x2C   | TIMx_ARR      |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reserved      |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x30   | Reserved      |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x34   | TIMx_CCR1     |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x38   | TIMx_CCR2     |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x3C to 0x64 | Reserved |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x68   | TIM12_TISEL   |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.3 on page 128 for the register boundary addresses.
44.5 TIM13/TIM14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

44.5.1 TIMx control register 1 (TIMx_CR1)(x = 13 to 14)

Address offset: 0x00

Reset value: 0x0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>UIFREMAP</th>
<th>CKD[1:0]</th>
<th>ARPE</th>
<th></th>
<th></th>
<th></th>
<th>OPM</th>
<th>URS</th>
<th>UDIS</th>
<th>CEN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 UIFREMAP: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 CKD[1:0]: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (TIx),
00: tDTS = tCK_INT
01: tDTS = 2 × tCK_INT
10: tDTS = 4 × tCK_INT
11: Reserved

Bit 7 ARPE: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 OPM: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).
44.5.2  TIMx Interrupt enable register (TIMx_DIER)(x = 13 to 14)

Address offset: 0x0C
Reset value: 0x0000

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

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

Bit 1  CC1IE: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0  UIE: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

44.5.3  TIMx status register (TIMx_SR)(x = 13 to 14)

Address offset: 0x10
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
</tr>
</tbody>
</table>

Bit 2  URS: Update request source
This bit is set and cleared by software to select the update interrupt (UEV) sources.
0: Any of the following events generate an UEV if enabled:
  – Counter overflow
  – Setting the UG bit
1: Only counter overflow generates an UEV if enabled.

Bit 1  UDIS: Update disable
This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.
0: UEV enabled. An UEV is generated by one of the following events:
  – Counter overflow
  – Setting the UG bit.
Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0  CEN: Counter enable
0: Counter disabled
1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.
Bits 15:10  Reserved, must be kept at reset value.

Bit 9  CC1OF: Capture/Compare 1 overcapture flag
   This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.
   0: No overcapture has been detected.
   1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:2  Reserved, must be kept at reset value.

Bit 1  CC1IF: Capture/compare 1 interrupt flag
   This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
   0: No compare match / No input capture occurred
   1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0  UIF: Update interrupt flag
   This bit is set by hardware on an update event. It is cleared by software.
   0: No update occurred.
   1: Update interrupt pending. This bit is set by hardware when the registers are updated:
      – At overflow and if UDIS='0' in the TIMx_CR1 register.
      – When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS='0' and UDIS='0' in the TIMx_CR1 register.

44.5.4  TIMx event generation register (TIMx_EGR)(x = 13 to 14)

Address offset: 0x14

Reset value: 0x0000
44.5.5 TIMx capture compare mode register 1 [alternate]

(TIMx_CCMR1)(x = 13 to 14)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Input capture mode:**

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

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

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at fDTS
0001: fsampling=fCK_INT, N=21001: fsampling=fDTS/8, N=8
0010: fsampling=fCK_INT, N=41010: fsampling=fDTS/16, N=5
0011: fsampling=fCK_INT, N=81011: fsampling=fDTS/32, N=6
0100: fsampling=fDTS/2, N=61100: fsampling=fDTS/16, N=8
0101: fsampling=fDTS/2, N=81101: fsampling=fDTS/32, N=5
0110: fsampling=fDTS/4, N=61110: fsampling=fDTS/32, N=6
0111: fsampling=fDTS/4, N=81111: fsampling=fDTS/32, N=8

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:

*If channel CC1 is configured as output:*
CC1IF flag is set, corresponding interrupt or is sent if enabled.

*If channel CC1 is configured as input:*
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.
### 44.5.6 TIMx capture/compare mode register 1 [alternate]

**TIMx_CCMR1(x = 13 to 14)**

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

#### Output compare mode:

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

**Note:** CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).
Bits 31:17  Reserved, must be kept at reset value.

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

Bits 16, 6:4  **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas OC1 active level depends on CC1P bit.

0000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active

Others: Reserved

**Note:** In PWM mode 1 or 2, the OCREF level changes when the result of the comparison changes or when the output compare mode switches from frozen to PWM mode.

Bit 3  **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

**Note:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2  **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. OC is then set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0  **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: Reserved.

11: Reserved.

**Note:** CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).
44.5.7 TIMx capture/compare enable register
(TIMx_CCER)(x = 13 to 14)

Address offset: 0x20
Reset value: 0x0000

Bits 15:4 Reserved, must be kept at reset value.

Bit 3  CC1NP: Capture/Compare 1 complementary output Polarity.
CC1 channel configured as output: CC1NP must be kept cleared.
CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define
TI1FP1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1  CC1P: Capture/Compare 1 output Polarity.
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active
polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge
(capture or trigger operations in reset, external clock or trigger mode),
TIxFP1 is not inverted (trigger operation in gated mode or encoder
mode).
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge
(capture or trigger operations in reset, external clock or trigger mode),
TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising
and falling edges (capture or trigger operations in reset, external clock
or trigger mode), TIxFP1is not inverted (trigger operation in gated
mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=0: This configuration is reserved, it must not be used.

Bit 0  CC1E: Capture/Compare 1 output enable.
0: Capture mode disabled / OC1 is not active
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 316. Output control bit for standard OCx channels

<table>
<thead>
<tr>
<th>CCxE bit</th>
<th>OCx output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output Disabled (OCx='0', OCx_EN='0')</td>
</tr>
<tr>
<td>1</td>
<td>OCx=OCxREF + Polarity, OCx_EN='1'</td>
</tr>
</tbody>
</table>

Note: The state of the external I/O pins connected to the standard OCx channels depends on the
OCx channel state and the GPIO registers.
44.5.8 TIMx counter (TIMx_CNT)(x = 13 to 14)
Address offset: 0x24
Reset value: 0x0000 0000

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

44.5.9 TIMx prescaler (TIMx_PSC)(x = 13 to 14)
Address offset: 0x28
Reset value: 0x0000

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency CK_CNT is equal to fCK_PSC / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event.
(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

44.5.10 TIMx auto-reload register (TIMx_ARR)(x = 13 to 14)
Address offset: 0x2C
Reset value: 0xFFFF

Bits 15:0 **ARR[15:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to Section 44.3.1: Time-base unit on page 1623 for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
44.5.11  **TIMx capture/compare register 1 (TIMx_CCR1)** \((x = 13 \text{ to } 14)\)

Address offset: 0x34

Reset value: 0x0000

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

Bits 15:0  **CCR1[15:0]**: Capture/Compare 1 value

**If channel CC1 is configured as output:**

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

**If channel CC1 is configured as input:**

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

44.5.12  **TIM13 timer input selection register (TIM13_TISEL)**

Address offset: 0x68

Reset value: 0x0000

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

Bits 15:4  **Reserved**, must be kept at reset value.

Bits 3:0  **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

- 0000: TIM13_CH1 input
- Other: Reserved

44.5.13  **TIM14 timer input selection register (TIM14_TISEL)**

Address offset: 0x68

Reset value: 0x0000

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

Bits 15:4  **Reserved**, must be kept at reset value.

Bits 3:0  **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

- 0000: TIM14_CH1 input
- Other: Reserved
### 44.5.14 TIM13/TIM14 register map

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

#### Table 317. TIM13/TIM14 register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>TIMx_CR1</td>
<td>0x04</td>
<td>Reserved</td>
<td>0x08</td>
<td>Reserved</td>
<td>0x0C</td>
<td>TIMx_DIER</td>
<td>0x10</td>
<td>TIMx_SR</td>
<td>0x14</td>
<td>TIMx_EGR</td>
<td>0x18</td>
<td>TIMx_CCMR1</td>
<td>0x1C</td>
<td>Reserved</td>
<td>0x20</td>
<td>TIMx_CCER</td>
<td>0x24</td>
<td>TIMx_CNT</td>
<td>0x28</td>
<td>TIMx_PSC</td>
<td>0x2C</td>
<td>TIMx_ARR</td>
<td>0x30</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00</td>
<td>TIMx_CR1</td>
<td>0x04</td>
<td>Reserved</td>
<td>0x08</td>
<td>Reserved</td>
<td>0x0C</td>
<td>TIMx_DIER</td>
<td>0x10</td>
<td>TIMx_SR</td>
<td>0x14</td>
<td>TIMx_EGR</td>
<td>0x18</td>
<td>TIMx_CCMR1</td>
<td>0x1C</td>
<td>Reserved</td>
<td>0x20</td>
<td>TIMx_CCER</td>
<td>0x24</td>
<td>TIMx_CNT</td>
<td>0x28</td>
<td>TIMx_PSC</td>
<td>0x2C</td>
<td>TIMx_ARR</td>
<td>0x30</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00</td>
<td>TIMx_CR1</td>
<td>0x04</td>
<td>Reserved</td>
<td>0x08</td>
<td>Reserved</td>
<td>0x0C</td>
<td>TIMx_DIER</td>
<td>0x10</td>
<td>TIMx_SR</td>
<td>0x14</td>
<td>TIMx_EGR</td>
<td>0x18</td>
<td>TIMx_CCMR1</td>
<td>0x1C</td>
<td>Reserved</td>
<td>0x20</td>
<td>TIMx_CCER</td>
<td>0x24</td>
<td>TIMx_CNT</td>
<td>0x28</td>
<td>TIMx_PSC</td>
<td>0x2C</td>
<td>TIMx_ARR</td>
<td>0x30</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Refer to *Section 2.3 on page 128* for the register boundary addresses.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x34</td>
<td>TIMx_CCR1</td>
<td>0x38</td>
<td>Reserved</td>
<td>0x68</td>
<td>TIM13_TISEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x68</td>
<td>Reserved</td>
<td>0x68</td>
<td>TIM14_TISEL</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td>0 0 0 0</td>
<td></td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIMx_CCR1</td>
<td>0x38</td>
<td>Reserved</td>
<td>0x68</td>
<td>TIM13_TISEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x68</td>
<td>Reserved</td>
<td>0x68</td>
<td>TIM14_TISEL</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td>0 0 0 0</td>
<td></td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

Refer to *Section 2.3 on page 128* for the register boundary addresses.
45 General-purpose timers (TIM15/TIM16/TIM17)

45.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in Section 45.4.22: Timer synchronization (TIM15).

45.2 TIM15 main features

TIM15 includes the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)
45.3 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input
1. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to Section 8.5.3: Clock security system (CSS)
   - A PVD output
   - all SRAM and TCM double ECC errors
   - Flash double ECC error
   - Cortex®-M7 LOCKUP (Hardfault) output
   - COMP output
1. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to Section 8.5.3: Clock security system (CSS)
   - A PVD output
   - all SRAM and TCM double ECC errors
   - Flash double ECC error
   - Cortex®-M7 LOCKUP (Hardfault) output
   - COMP output
45.4 **TIM15/TIM16/TIM17 functional description**

45.4.1 **Time-base unit**

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- **Counter register (TIMx_CNT)**
- **Prescaler register (TIMx_PSC)**
- **Auto-reload register (TIMx_ARR)**
- **Repetition counter register (TIMx_RCR)**

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

**Prescaler description**

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 450 and Figure 451* give some examples of the counter behavior when the prescaler ratio is changed on the fly:
Figure 450. Counter timing diagram with prescaler division change from 1 to 2

CK_PSC

CEN

Timer clock = CK_CNT

Counter register: F7 F8 F9 FA FB FC 00 01 02 03

Update event (UEV)

Prescaler control register: 0 1

Write a new value in TIMx_PSC

Prescaler buffer: 0 1

Prescaler counter: 0 0 1 0 1 0 1 0 1

Figure 451. Counter timing diagram with prescaler division change from 1 to 4

CK_PSC

CEN

Timer clock = CK_CNT

Counter register: F7 F8 F9 FA FB FC 00 01

Update event (UEV)

Prescaler control register: 0 3

Write a new value in TIMx_PSC

Prescaler buffer: 0 3

Prescaler counter: 0 0 1 2 3 0 1 2 3
45.4.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.
Figure 452. Counter timing diagram, internal clock divided by 1

Figure 453. Counter timing diagram, internal clock divided by 2
Figure 454. Counter timing diagram, internal clock divided by 4

Figure 455. Counter timing diagram, internal clock divided by N
**Figure 456. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)**

- **CK_PSC**
- **CEN**
- **Timer clock = CK_CNT**
- **Counter register**
- **Counter overflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**
- **Auto-reload preload register**

Write a new value in TIMx.ARR

**Figure 457. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)**

- **CK_PSC**
- **CEN**
- **Timer clock = CK_CNT**
- **Counter register**
- **Counter overflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**
- **Auto-reload preload register**
- **Auto-reload shadow register**

Write a new value in TIMx.ARR
45.4.3 Repetition counter

Section 45.4.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to Figure 458). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.
45.4.4 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin
- Internal trigger inputs (ITRx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to Using one timer as prescaler for another timer on page 1584 for more details.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed...
only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

*Figure 459* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

*Figure 459. Control circuit in normal mode, internal clock divided by 1*

![Control circuit in normal mode, internal clock divided by 1](image)

**External clock source mode 1**

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

*Figure 460. TI2 external clock connection example*

![TI2 external clock connection example](image)

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:
1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.

2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ’01’ in the TIMx_CCMR1 register.

3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).

4. Select rising edge polarity by writing CC2P=0 in the TIMx_CCR1 register.

5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.

6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.

7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so it does not need to be configured.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

**Figure 461. Control circuit in external clock mode 1**

45.4.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

*Figure 462 to Figure 465* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).
The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 462. Capture/compare channel (example: channel 1 input stage)

Figure 463. Capture/compare channel 1 main circuit
The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

45.4.6 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was
already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to ‘0’ or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when Ti1 input rises. To do this, use the following procedure:

1. Select the proper Ti1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the Ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TiRx (ICxF bits in the TIMx_CCMRx register). Let’s imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on Ti1 when 8 consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the Ti1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:
- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

45.4.7 PWM input mode (only for TIM15)

This mode is a particular case of input capture mode. The procedure is the same except:
- Two ICx signals are mapped on the same TiRx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TiFxP signals is selected as trigger input and the slave mode controller is configured in reset mode.
For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper TI1[x] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to ‘1’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

**Figure 466. PWM input mode timing**

1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

**45.4.8 Forced output mode**

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCXREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.
Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

45.4.9 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxiE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 467.
45.4.10 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘110’ (PWM mode 1) or ‘111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to Upcounting mode on page 1678.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at

---

**Figure 467. Output compare mode, toggle on OC1**

Write B201h in the CC1R register

Match detected on CCR1
Interrupt generated if enabled

OC1REF = OC1

TIM1_CNT: 0039 003A 003B B200 B201
TIM1_CCR1: 003A B201

Write B201h in the CC1R register

Match detected on CCR1
Interrupt generated if enabled
‘1’. If the compare value is 0 then OCxRef is held at ‘0’. Figure 468 shows some edge-aligned PWM waveforms in an example where TIMx.ARR=8.

**Figure 468. Edge-aligned PWM waveforms (ARR=8)**

<table>
<thead>
<tr>
<th>Counter register</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCRx=4 OCXREF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx=8 OCXREF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx&gt;8 OCXREF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx=0 OCXREF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**45.4.11 Combined PWM mode (TIM15 only)**

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx.ARR register, the duty cycle and delay are determined by the two TIMx.CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by the TIMx.CCR1 and TIMx.CCR2 registers

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx.CCMRx register.

When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

**Note:** The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

**Figure 469** represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
45.4.12 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to Table 322: Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) on page 1748 for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a
reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

**Figure 470. Complementary output with dead-time insertion.**

**Figure 471. Dead-time waveforms with delay greater than the negative pulse.**
The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to Section 45.6.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 1751 for delay calculation.

**Re-directing OCxREF to OCx or OCxN**

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

*Note:* When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

**45.4.13 Using the break function**

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM15/TIM16/TIM17 timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.
The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows to enable /disable the outputs by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shutdown level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to Table 320: Output control bits for complementary OCx and OCxN channels with break feature (TIM15) on page 1726 for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

A programmable filter (BKF[3:0] bits in the TIMx_BDTR register allows to filter out spurious events.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_OR2 register.

The sources for break (BRK) channel are:

- An external source connected to one of the BKin pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source:
  - the output from a comparator, with polarity selection and optional digital filtering
  - the analog watchdog output of the DFSDM1 peripheral
  - A system break:
    - the Cortex®-M7 LOCKUP output
    - the PVD output
    - all SRAM and TCM double ECC errors (AXI-SRAM, ITCM, DTCM, SRAM1, SRAM2, SRAM3, SRAM4, BKRAM, refer to SYSCFG_CFGR register for details)
    - a Flash double ECC error
    - a clock failure event generated by the CSS detector
Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the AFIO controller (selected by the OSSi bit). This feature functions even if the MCU oscillator is off.

- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSi=0, the timer releases the output control (taken over by the AFIO controller) else the enable output remains high.

- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their...
active level together. Note that because of the resynchronization on MOE, the
dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).

- If OSS1=0 then the timer releases the enable outputs (taken over by the AFIO
controller which forces a Hi-Z state) else the enable outputs remain or become
high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be
generated if the BIE bit in the TIMx_DIER register is set.

- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again
at the next update event UEV. This can be used to perform a regulation, for instance.
Else, MOE remains low until it is written with 1 again. In this case, it can be used for
security and the break input can be connected to an alarm from power drivers, thermal
sensors or any security components.

**Note:** The break inputs is acting on level. Thus, the MOE cannot be set while the break input is
active (neither automatically nor by software). In the meantime, the status flag BIF cannot
be cleared.

The break can be generated by the BRK input which has a programmable polarity and an
enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been
implemented inside the break circuit to safeguard the application. It allows the configuration
of several parameters to be frozen (dead-time duration, OCx/OCxN polarities and state
when disabled, OCxM configurations, break enable and polarity). The protection can be
selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to
Section 45.6.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on
page 1751. The LOCK bits can be written only once after an MCU reset.

The Figure 474 shows an example of behavior of the outputs in response to a break.
Figure 474. Output behavior in response to a break

OCxREF

OCx (OCxN not implemented, CCxP=0, OISx=1)

OCx (OCxN not implemented, CCxP=0, OISx=0)

OCx (OCxN not implemented, CCxP=1, OISx=1)

OCx (OCxN not implemented, CCxP=1, OISx=0)

OCx

OCxN delay delay delay

(OCxE=1, CCxP=0, OISx=0, CCxNE=1, CCxNP=0, OISxN=1)

OCxN delay delay delay

(CCxE=1, CCxP=0, OISx=1, CCxNE=1, CCxNP=1, OISxN=1)

OCxN delay delay delay

(CCxE=1, CCxP=0, OISx=0, CCxNE=1, CCxNP=1, OISxN=1)

OCxN delay

(CCxE=1, CCxP=0, CCxNE=1, CCxNP=0, OISxN=1)

OCxN delay

(CCxE=1, CCxP=0, CCxNE=0, CCxNP=0, OISxN=0)

OCxN delay

(CCxE=1, CCxP=0, CCxNE=0, CCxNP=0, OISxN=0 or OISx=OISxN=1)
45.4.14  Bidirectional break inputs

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on Figure 475.

They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (BG) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see Table 318)

<table>
<thead>
<tr>
<th>MOE</th>
<th>BKDIR</th>
<th>BKDSRM</th>
<th>Break protection state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Disarmed</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Armed</td>
</tr>
</tbody>
</table>

Arming and re-arming break circuitry

The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).
The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

**Figure 475. Output redirection**
45.4.15 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- \( CNT < CCRx \leq ARR \) (in particular, \( 0 < CCRx \))
For example one may want to generate a positive pulse on OC1 with a length of $t_{PULSE}$ and after a delay of $t_{DELAY}$ as soon as a positive edge is detected on the TI2 input pin.

Let’s use TI2FP2 as trigger 1:

1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='00110' in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to ‘110’ in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The $t_{DELAY}$ is defined by the value written in the TIMx_CCR1 register.
- The $t_{PULSE}$ is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).

Let’s say one want to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0’ in this example.

Since only 1 pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).
Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay $t_{\text{DELAY min}}$ we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking into account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

45.4.16 Retriggerable one pulse mode (OPM) (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in Section 45.4.15:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

45.4.17 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows both
the counter value and a potential roll-over condition signaled by the UIFCPY flag, to be atomically read. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.
45.4.18 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins TIMx_CH1 and TIMx_CH2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in Figure 478.

Figure 478. Measuring time interval between edges on 2 signals
45.4.19 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining. The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P='0' and CC1NP='0' in the TIMx_CCRER register to validate the polarity (and detect rising edges only).

2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

![Figure 479. Control circuit in reset mode](MS31401V1)
Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP = ‘0’ in the TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 480. Control circuit in gated mode

![Control circuit in gated mode](MS31402V1)
**Slave mode: Trigger mode**

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P='1' and CC2NP='0' in the TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS=110 in the TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in the TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

![Control circuit in trigger mode](MS31403V1)

**45.4.20 Slave mode – combined reset + trigger mode**

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

**45.4.21 DMA burst mode**

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to re-program several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.
The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address, i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,

For example, the timer DMA burst feature could be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   – Number of data to transfer = 3 (See note below).
   – Circular mode disabled.

2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:
   – DBL = 3 transfers, DBA = 0xE.

3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).

4. Enable TIMx

5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.
45.4.22 Timer synchronization (TIM15)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to Section 43.3.19: Timer synchronization for details.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

45.4.23 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter either continues to work normally or stops, depending on TIMx bit in DBGMCU module. For more details, refer to Section 60: Debug infrastructure.

For safety purposes, when the counter is stopped (TIMx = 1 in DBGMCU_APB2FZ1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.
45.5 TIM15 registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.
The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

45.5.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x000
Reset value: 0x0000

<table>
<thead>
<tr>
<th></th>
<th>UIFRE</th>
<th>CKD[1:0]</th>
<th>ARPE</th>
<th>OPM</th>
<th>URS</th>
<th>UDIS</th>
<th>CEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
- 0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
- 1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
- This bitfield indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (tDTS) used by the dead-time generators and the digital filters (TIx)
- 00: tDTS = tCK_INT
- 01: tDTS = 2*tCK_INT
- 10: tDTS = 4*tCK_INT
- 11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable
- 0: TIMx_ARR register is not buffered
- 1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
- 0: Counter is not stopped at update event
- 1: Counter stops counting at the next update event (clearing the bit CEN)
Bit 2 **URS**: Update request source
- This bit is set and cleared by software to select the UEV event sources.
- 0: Any of the following events generate an update interrupt if enabled. These events can be:
  - Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller
- 1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable
- This bit is set and cleared by software to enable/disable UEV event generation.
- 0: UEV enabled. The Update (UEV) event is generated by one of the following events:
  - Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller
- Buffered registers are then loaded with their preload values.
- 1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
- 0: Counter disabled
- 1: Counter enabled

*Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

### 45.5.2 TIM15 control register 2 (TIM15_CR2)

**Address offset: 0x04**

**Reset value: 0x0000**

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

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

**Bit 10 OIS2**: Output idle state 2 (OC2 output)
- 0: OC2=0 when MOE=0
- 1: OC2=1 when MOE=0

*Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15_BDTR register).*

**Bit 9 OIS1N**: Output Idle state 1 (OC1N output)
- 0: OC1N=0 after a dead-time when MOE=0
- 1: OC1N=1 after a dead-time when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).*

**Bit 8 OIS1**: Output Idle state 1 (OC1 output)
- 0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
- 1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).*
Bit 7  **TI1S**: TI1 selection
   0: The TIMx_CH1 pin is connected to TI1 input
   1: The TIMx_CH1, CH2 pins are connected to the TI1 input (XOR combination)

Bits 6:4  **MMS[2:0]**: Master mode selection
   These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
   000: Reset - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
   001: Enable - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
   010: Update - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
   011: Compare Pulse - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred.
   100: Compare - OC1REF signal is used as trigger output (TRGO).
   101: Compare - OC2REF signal is used as trigger output (TRGO).

Bit 3  **CCDS**: Capture/compare DMA selection
   0: CCx DMA request sent when CCx event occurs
   1: CCx DMA requests sent when update event occurs

Bit 2  **CCUS**: Capture/compare control update selection
   0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.
   1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.
   *Note*: This bit acts only on channels that have a complementary output.

Bit 1  Reserved, must be kept at reset value.

Bit 0  **CCPC**: Capture/compare preloaded control
   0: CCxE, CCxNE and OCxM bits are not preloaded
   1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).
   *Note*: This bit acts only on channels that have a complementary output.
### 45.5.3 TIM15 slave mode control register (TIM15_SMCR)

Address offset: 0x08  
Reset value: 0x0000 0000

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

Bits 31:22: Reserved, must be kept at reset value.  
Bits 19:17: Reserved, must be kept at reset value.  
Bits 15:8: Reserved, must be kept at reset value.  

Bit 7 **MSM:** Master/slave mode  
0: No action  
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection  
This bit field selects the trigger input to be used to synchronize the counter.  
00000: Internal Trigger 0 (ITR0)  
00001: Internal Trigger 1 (ITR1)  
00010: Internal Trigger 2 (ITR2)  
00011: Internal Trigger 3 (ITR3)  
00100: TI1 Edge Detector (TI1F_ED)  
00101: Filtered Timer Input 1 (TI1FP1)  
00110: Filtered Timer Input 2 (TI2FP2)  
Other: Reserved  
See [Table 319: TIMx Internal trigger connection on page 1716](#) for more details on ITRx meaning for each Timer.  

**Note:** These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3: Reserved, must be kept at reset value.
Bits 16, 2, 1, 0  **SMS[3:0]**: Slave mode selection

- When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description).
- 0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.
- 0001: Reserved
- 0010: Reserved
- 0011: Reserved
- 0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
- 0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
- 0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
- 0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
- 1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.
- Other codes: reserved.

Note: The gated mode must not be used if Ti1F_ED is selected as the trigger input (TS=’00100’). Indeed, Ti1F_ED outputs 1 pulse for each transition on Ti1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

### Table 319. TIMx Internal trigger connection

<table>
<thead>
<tr>
<th>Slave Tim</th>
<th>ITR0 (TS = 00000)</th>
<th>ITR1 (TS = 00001)</th>
<th>ITR2 (TS = 00010)</th>
<th>ITR3 (TS = 00011)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM15</td>
<td>TIM1</td>
<td>TIM3</td>
<td>TIM16 OC1</td>
<td>TIM17 OC1</td>
</tr>
</tbody>
</table>

### 45.5.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>TDE</td>
<td>COMD</td>
<td>E</td>
<td>Res</td>
<td>Res</td>
<td>CC2DE</td>
<td>CC1DE</td>
<td>UDE</td>
<td>BIE</td>
<td>TIE</td>
<td>COMIE</td>
<td>Res</td>
<td>Res</td>
<td>CC2IE</td>
<td>CC1IE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

- Bit 15  **Reserved**, must be kept at reset value.
- Bit 14  **TDE**: Trigger DMA request enable
  0: Trigger DMA request disabled
  1: Trigger DMA request enabled
- Bit 13  **COMDE**: COM DMA request enable
  0: COM DMA request disabled
  1: COM DMA request enabled
Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
- 0: CC2 DMA request disabled
- 1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
- 0: CC1 DMA request disabled
- 1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
- 0: Update DMA request disabled
- 1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
- 0: Break interrupt disabled
- 1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
- 0: Trigger interrupt disabled
- 1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
- 0: COM interrupt disabled
- 1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
- 0: CC2 interrupt disabled
- 1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
- 0: CC1 interrupt disabled
- 1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
- 0: Update interrupt disabled
- 1: Update interrupt enabled

### 45.5.5 TIM15 status register (TIM15_SR)

**Address offset:** 0x10

**Reset value:** 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CC2OF</td>
<td>CC1OF</td>
<td>BIF</td>
<td>TIF</td>
<td>COMIF</td>
<td></td>
<td>CC2IF</td>
<td>CC1IF</td>
<td>UIF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td></td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td></td>
</tr>
</tbody>
</table>
Bits 15:11  Reserved, must be kept at reset value.

Bit 10  **CC2OF**: Capture/Compare 2 overcapture flag  
Refer to CC1OF description

Bit 9  **CC1OF**: Capture/Compare 1 overcapture flag  
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
- 0: No overcapture has been detected
- 1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8  Reserved, must be kept at reset value.

Bit 7  **BIF**: Break interrupt flag  
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
- 0: No break event occurred
- 1: An active level has been detected on the break input

Bit 6  **TIF**: Trigger interrupt flag  
This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
- 0: No trigger event occurred
- 1: Trigger interrupt pending

Bit 5  **COMIF**: COM interrupt flag  
This flag is set by hardware on a COM event (once the capture/compare control bits —CCxE, CCxNE, OCxM— have been updated). It is cleared by software.
- 0: No COM event occurred
- 1: COM interrupt pending

Bits 4:3  Reserved, must be kept at reset value.
45.5.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14
Reset value: 0x1000

<table>
<thead>
<tr>
<th>Bit 15:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7</td>
<td>BG: Break generation</td>
</tr>
<tr>
<td></td>
<td>This bit is set by software in order</td>
</tr>
<tr>
<td></td>
<td>to generate an event, it is</td>
</tr>
<tr>
<td></td>
<td>automatically cleared by</td>
</tr>
<tr>
<td></td>
<td>hardware.</td>
</tr>
<tr>
<td></td>
<td>0: No action</td>
</tr>
<tr>
<td></td>
<td>1: A break event is generated. MOE</td>
</tr>
<tr>
<td></td>
<td>bit is cleared and BIF flag is set.</td>
</tr>
<tr>
<td></td>
<td>Related interrupt or DMA transfer can</td>
</tr>
<tr>
<td></td>
<td>occur if enabled.</td>
</tr>
<tr>
<td>Bit 6</td>
<td>TG: Trigger generation</td>
</tr>
<tr>
<td></td>
<td>This bit is set by software in order</td>
</tr>
<tr>
<td></td>
<td>to generate an event, it is</td>
</tr>
<tr>
<td></td>
<td>automatically cleared by</td>
</tr>
<tr>
<td></td>
<td>hardware.</td>
</tr>
<tr>
<td></td>
<td>0: No action</td>
</tr>
<tr>
<td></td>
<td>1: The TIF flag is set in TIMx_SR</td>
</tr>
<tr>
<td></td>
<td>register. Related interrupt or DMA</td>
</tr>
<tr>
<td></td>
<td>transfer can occur if enabled.</td>
</tr>
</tbody>
</table>
Bit 5 **COMG**: Capture/Compare control update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

*Note*: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation
Refer to CC1G description.

Bit 1 **CC1G**: Capture/Compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:

- **If channel CC1 is configured as output**: CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
- **If channel CC1 is configured as input**: The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

### 45.5.7 TIM15 capture/compare mode register 1 [alternate]
(TIM15_CCMR1)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Input capture mode**:

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

- **IC2F[3:0]**: Input capture 2 filter
- **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler
Bits 9:8  **CC2S[1:0]: Capture/Compare 2 selection**

This bit-field defines the direction of the channel (input/output) as well as the used input.

- 00: CC2 channel is configured as output
- 01: CC2 channel is configured as input, IC2 is mapped on TI2
- 10: CC2 channel is configured as input, IC2 is mapped on TI1
- 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).*

Bits 7:4  **IC1F[3:0]: Input capture 1 filter**

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at f_{DTS}
- 0001: f_{SAMPLING}=f_{CK_INT}, N=2
- 0010: f_{SAMPLING}=f_{CK_INT}, N=4
- 0011: f_{SAMPLING}=f_{CK_INT}, N=8
- 0100: f_{SAMPLING}=f_{DTS}/2, N=6
- 0101: f_{SAMPLING}=f_{DTS}/2, N=8
- 0110: f_{SAMPLING}=f_{DTS}/4, N=6
- 0111: f_{SAMPLING}=f_{DTS}/4, N=8
- 1000: f_{SAMPLING}=f_{DTS}/8, N=6
- 1001: f_{SAMPLING}=f_{DTS}/8, N=8
- 1010: f_{SAMPLING}=f_{DTS}/16, N=5
- 1011: f_{SAMPLING}=f_{DTS}/16, N=6
- 1100: f_{SAMPLING}=f_{DTS}/32, N=5
- 1101: f_{SAMPLING}=f_{DTS}/32, N=6
- 1110: f_{SAMPLING}=f_{DTS}/32, N=8
- 1111: f_{SAMPLING}=f_{DTS}/32, N=8

Bits 3:2  **IC1PSC[1:0]: Input capture 1 prescaler**

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0  **CC1S[1:0]: Capture/Compare 1 Selection**

This bit-field defines the direction of the channel (input/output) as well as the used input.

- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, IC1 is mapped on TI1
- 10: CC1 channel is configured as input, IC1 is mapped on TI2
- 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).*

### 45.5.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the
corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Output compare mode:**

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| OC2M[2:0] | OC2 PE | OC2 FE | CC2S[1:0] | rw | OC1M[2:0] | OC1 PE | OC1 FE | CC1S[1:0] |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 24, 14:12 **OC2M[3:0]**: Output Compare 2 mode

- Bit 11 **OC2PE**: Output Compare 2 preload enable
- Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- 00: CC2 channel is configured as output.
- 01: CC2 channel is configured as input, IC2 is mapped on TI2.
- 10: CC2 channel is configured as input, IC2 is mapped on TI1.
- 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register).

*Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).*

Bit 7 Reserved, must be kept at reset value.
Bits 16, 6:4 **OC1M[3:0]: Output Compare 1 mode**

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Reserved

1111: Reserved,

**Note:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels that have a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.
General-purpose timers (TIM15/TIM16/TIM17)

Bit 3 **OC1PE**: Output Compare 1 preload enable

- **0**: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
- **1**: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note*: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

- **0**: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
- **1**: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- **00**: CC1 channel is configured as output.
- **01**: CC1 channel is configured as input, IC1 is mapped on TI1.
- **10**: CC1 channel is configured as input, IC1 is mapped on TI2.
- **11**: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note*: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

### 45.5.9 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC2NP</td>
<td>CC2P</td>
<td>CC2E</td>
<td>CC1NP</td>
<td>CC1NE</td>
<td>CC1P</td>
<td>CC1E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:8 Reserved, must be kept at reset value.

- **Bit 7 CC2NP**: Capture/Compare 2 complementary output polarity
  - Refer to CC1NP description

- **Bit 6 Reserved, must be kept at reset value.**

- **Bit 5 CC2P**: Capture/Compare 2 output polarity
  - Refer to CC1P description

- **Bit 4 CC2E**: Capture/Compare 2 output enable
  - Refer to CC1E description
Bit 3  **CC1NP**: Capture/Compare 1 complementary output polarity

- CC1 channel configured as output:
  0: OC1N active high
  1: OC1N active low

- CC1 channel configured as input:
  This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

*Note:* This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2  **CC1NE**: Capture/Compare 1 complementary output enable

- 0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
- 1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1  **CC1P**: Capture/Compare 1 output polarity

- 0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
- 1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

*When CC1 channel is configured as input*, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

- CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).
- CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
- CC1NP=1, CC1P=0: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
- CC1NP=1, CC1P=1: this configuration is reserved, it must not be used.

*Note:* This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0  **CC1E**: Capture/Compare 1 output enable

- 0: Capture mode disabled / OC1 is not active (see below)
- 1: Capture mode enabled / OC1 signal is output on the corresponding output pin

*When CC1 channel is configured as output*, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to Table 320 for details.
Table 320. Output control bits for complementary OCx and OCxN channels with break feature (TIM15)

<table>
<thead>
<tr>
<th>Control bits</th>
<th>MOE bit</th>
<th>OSSI bit</th>
<th>OSSR bit</th>
<th>CCxE bit</th>
<th>CCxNE bit</th>
<th>OCx output state</th>
<th>OCxN output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Output Disabled (not driven by the timer: Hi-Z)</td>
<td>OCx=0 OCxN=0, OCxN_EN=0</td>
</tr>
<tr>
<td>01</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Output Disabled (not driven by the timer: Hi-Z)</td>
<td>OCx=0 OCxN=OCxREF XOR CCxNP</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>OCREF + Polarity + dead-time</td>
<td>Complementary to OCREF (not OCREF) + Polarity + dead-time</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Off-State (output enabled with inactive state)</td>
<td>OCx=OCxREF XOR CCxP OCxN=OCxREF XOR CCxNP</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Output disabled (not driven by the timer anymore). The output state is defined by the GPIO controller and can be High, Low or Hi-Z.</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP</td>
<td></td>
</tr>
</tbody>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and AFIO registers.
45.5.10 TIM15 counter (TIM15_CNT)

Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UIFCPY: UIF Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>UIF Copy</td>
</tr>
</tbody>
</table>

This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 CNT[15:0]: Counter value

45.5.11 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>PSC[15:0]: Prescaler value</th>
</tr>
</thead>
</table>

The counter clock frequency (CK_CNT) is equal to fCK_PSC / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in "reset mode").

45.5.12 TIM15 auto-reload register (TIM15_ARR)

Address offset: 0x2C
Reset value: 0xFFFF

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>ARR[15:0]: Auto-reload value</th>
</tr>
</thead>
</table>

ARR is the value to be loaded in the actual auto-reload register.

Refer to the Section 45.4.1: Time-base unit on page 1676 for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.
45.5.13 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits 15:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td><strong>REP[7:0]</strong></td>
</tr>
<tr>
<td></td>
<td>Repetition counter value</td>
</tr>
<tr>
<td></td>
<td>These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable. Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event. It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.</td>
</tr>
</tbody>
</table>

45.5.14 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th><strong>CCR1[15:0]</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Capture/Compare 1 value</td>
</tr>
<tr>
<td></td>
<td><strong>If channel CC1 is configured as output:</strong></td>
</tr>
<tr>
<td></td>
<td>CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.</td>
</tr>
<tr>
<td></td>
<td><strong>If channel CC1 is configured as input:</strong></td>
</tr>
<tr>
<td></td>
<td>CCR1 is the counter value transferred by the last input capture 1 event (IC1).</td>
</tr>
</tbody>
</table>
45.5.15  TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38
Reset value: 0x0000

<table>
<thead>
<tr>
<th>bits 15:0 CCR2[15:0]: Capture/Compare 2 value</th>
</tr>
</thead>
<tbody>
<tr>
<td>If channel CC2 is configured as output:</td>
</tr>
<tr>
<td>CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).</td>
</tr>
<tr>
<td>It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.</td>
</tr>
<tr>
<td>The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.</td>
</tr>
<tr>
<td>If channel CC2 is configured as input:</td>
</tr>
<tr>
<td>CCR2 is the counter value transferred by the last input capture 2 event (IC2).</td>
</tr>
</tbody>
</table>

45.5.16  TIM15 break and dead-time register (TIM15_BDTR)

Address offset: 0x44
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>bits 31:29 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 28 BKBID: Break Bidirectional</td>
</tr>
<tr>
<td>0: Break input BRK in input mode</td>
</tr>
<tr>
<td>1: Break input BRK in bidirectional mode</td>
</tr>
<tr>
<td>In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.</td>
</tr>
<tr>
<td>Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</td>
</tr>
<tr>
<td>Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.</td>
</tr>
</tbody>
</table>
Bit 27  Reserved, must be kept at reset value.

Bit 26  **BKDSRM**: Break Disarm
- 0: Break input BRK is armed
- 1: Break input BRK is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bits 25:20  Reserved, must be kept at reset value.

Bits 19:16  **BKF[3:0]**: Break filter
This bit-field defines the frequency used to sample the BRK input signal and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
- 0000: No filter, BRK acts asynchronously
- 0001: \( f_{\text{SAMPLING}} = f_{\text{CK_INT}} \), N=2
- 0010: \( f_{\text{SAMPLING}} = f_{\text{CK_INT}} \), N=4
- 0011: \( f_{\text{SAMPLING}} = f_{\text{CK_INT}} \), N=8
- 0100: \( f_{\text{SAMPLING}} = f_{\text{DTS}/2} \), N=6
- 0101: \( f_{\text{SAMPLING}} = f_{\text{DTS}/2} \), N=8
- 0110: \( f_{\text{SAMPLING}} = f_{\text{DTS}/4} \), N=6
- 0111: \( f_{\text{SAMPLING}} = f_{\text{DTS}/4} \), N=8
- 1000: \( f_{\text{SAMPLING}} = f_{\text{DTS}/8} \), N=6
- 1001: \( f_{\text{SAMPLING}} = f_{\text{DTS}/8} \), N=8
- 1010: \( f_{\text{SAMPLING}} = f_{\text{DTS}/16} \), N=5
- 1011: \( f_{\text{SAMPLING}} = f_{\text{DTS}/16} \), N=6
- 1100: \( f_{\text{SAMPLING}} = f_{\text{DTS}/16} \), N=8
- 1101: \( f_{\text{SAMPLING}} = f_{\text{DTS}/32} \), N=5
- 1110: \( f_{\text{SAMPLING}} = f_{\text{DTS}/32} \), N=6
- 1111: \( f_{\text{SAMPLING}} = f_{\text{DTS}/32} \), N=8

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 15  **MOE**: Main output enable
This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.
- 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
- 1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)
See OC/OCN enable description for more details (Section 45.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1724).

Bit 14  **AOE**: Automatic output enable
- 0: MOE can be set only by software
- 1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*
Bit 13 **BKP**: Break polarity
0: Break input BRK is active low
1: Break input BRK is active high

*Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable
0: Break inputs (BRK and CCS clock failure event) disabled
1: Break inputs (BRK and CCS clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode
This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 45.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1724*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)
1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

*Note: This bit cannot be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode
This bit is used when MOE=0 on channels configured as outputs.

See OC/OCN enable description for more details (*Section 45.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1724*).

0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)
1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1

*Note: This bit cannot be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration
These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

*Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.
Bits 7:0  **DTG[7:0]**: Dead-time generator setup
This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
- DTG[7:5]=0xx => DT=DTG[7:0]x tdtg with tdtg=tDTS
- DTG[7:5]=10x => DT=(64+DTG[5:0])x tdtg with tdtg=2xtDTS
- DTG[7:5]=110 => DT=(32+DTG[4:0])x tdtg with tdtg=8xtDTS
- DTG[7:5]=111 => DT=(32+DTG[4:0])x tdtg with tdtg=16xtDTS

Example if T\(_{DTS}\)=125ns (8MHz), dead-time possible values are:
- 0 to 15875 ns by 125 ns steps,
- 16 µs to 31750 ns by 250 ns steps,
- 32 µs to 63 µs by 1 µs steps,
- 64 µs to 126 µs by 2 µs steps

*Note*: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM\_x\_BDTR register).

### 45.5.17 TIM15 DMA control register (TIM15\_DCR)

Address offset: 0x48
Reset value: 0x0000

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

Bits 15:13  Reserved, must be kept at reset value.

Bits 12:8  **DBL[4:0]**: DMA burst length
This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM\_x\_DMAR address).
- 00000: 1 transfer,
- 00001: 2 transfers,
- 00010: 3 transfers,
- ...
- 10001: 18 transfers.

Bits 7:5  Reserved, must be kept at reset value.

Bits 4:0  **DBA[4:0]**: DMA base address
This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIM\_x\_DMAR address). DBA is defined as an offset starting from the address of the TIM\_x\_CR1 register.
Example:
- 00000: TIM\_x\_CR1,
- 00001: TIM\_x\_CR2,
- 00010: TIM\_x\_SMCR,
- ...

### 45.5.18 TIM15 DMA address for full transfer (TIM15\_DMAR)

Address offset: 0x4C
Reset value: 0x0000

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

1732/2974  RM0455 Rev 3
Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address 
(TIMx_CR1 address) + (DBA + DMA index) x 4 
where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base 
address configured in TIMx_DCR register, DMA index is automatically controlled by the 
DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

### 45.5.19 TIM15 alternate register 1 (TIM15_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bit 16</th>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:12 Reserved, must be kept at reset value.

**Bit 11 BKCMP2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP 
polarity bit.

- 0: COMP2 input is active low
- 1: COMP2 input is active high

*Note*: *This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

**Bit 10 BKCMP1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP 
polarity bit.

- 0: COMP1 input is active low
- 1: COMP1 input is active high

*Note*: *This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

**Bit 9 BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together 
with the BKP polarity bit.

- 0: BKIN input is active low
- 1: BKIN input is active high

*Note*: *This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

**Bit 8 BKDF1BK0E**: BRK dfsdm1_break[0] enable

This bit enables the dfsdm1_break[0] for the timer’s BRK input. dfsdm1_break[0] output is 
‘OREd’ with the other BRK sources.

- 0: dfsdm1_break[0]input disabled
- 1: dfsdm1_break[0]input enabled

*Note*: *This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bits 7:3 Reserved, must be kept at reset value.
Bit 2 **BKCMP2E**: BRK COMP2 enable
This bit enables the COMP2 for the timer’s BRK input. COMP2 output is ‘ORed’ with the other BRK sources.
0: COMP2 input disabled
1: COMP2 input enabled

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable
This bit enables the COMP1 for the timer’s BRK input. COMP1 output is ‘ORed’ with the other BRK sources.
0: COMP1 input disabled
1: COMP1 input enabled

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable
This bit enables the BKIN alternate function input for the timer’s BRK input. BKIN input is ‘ORed’ with the other BRK sources.
0: BKIN input disabled
1: BKIN input enabled

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

### 45.5.20 TIM15 input selection register (TIM15_TISEL)

Address offset: 0x68
Reset value: 0x0000 0000

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input
- 0000: TIM15_CH2 input
- 0001: TIM2_CH2 input
- 0010: TIM3_CH2 input
- 0011: TIM4_CH2 input
- Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.
45.5.21 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 3:0</th>
<th>Bits 15:4</th>
<th>Bits 23:16</th>
<th>Bits 31:24</th>
<th>Bits 47:40</th>
<th>Bits 63:56</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>TIM15_CR1</td>
<td>TI1SEL[3:0]</td>
<td>CKD[1:6]</td>
<td>APE</td>
<td>OP</td>
<td>CPM</td>
<td>URS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0000: TIM15_CH1 input</td>
<td>0001: TIM2_CH1 input</td>
<td>0010: TIM3_CH1 input</td>
<td>0011: TIM4_CH1 input</td>
<td>0100: LSE</td>
<td>0101: CSI</td>
</tr>
<tr>
<td>0x04</td>
<td>TIM15_CR2</td>
<td>OIS2</td>
<td>OIS1N</td>
<td>OIS1</td>
<td>TS[2:0]</td>
<td>SMS[2:0]</td>
<td>SMS[2:0]</td>
</tr>
<tr>
<td>0x0C</td>
<td>TIM15_DIER</td>
<td>TDE</td>
<td>COMDE</td>
<td>CDE</td>
<td>UDE</td>
<td>BIE</td>
<td>TIE</td>
</tr>
<tr>
<td>0x10</td>
<td>TIM15_SR</td>
<td>BF</td>
<td>TIF</td>
<td>COMIF</td>
<td>CC1FS</td>
<td>CC1E</td>
<td>CIE</td>
</tr>
<tr>
<td>0x14</td>
<td>TIM15_EGR</td>
<td>BG</td>
<td>TG</td>
<td>COMG</td>
<td>CC2FS</td>
<td>CC2E</td>
<td>CIE</td>
</tr>
</tbody>
</table>

Table 321. TIM15 register map and reset values
### Table 321. TIM15 register map and reset values (continued)

| Offset | Register name | Offset | Name                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  | Value                  |
|--------|---------------|--------|-----------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|
| 0x18   | TIM15_CCMR1   |        | OC2M[2:0]             | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      |
|        |               |        | CC1NS                  | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      |
| 0x20   | TIM15_CCER    |        | IC2F[3:0]             | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      |
| 0x24   | TIM15_CNT     |        | CNT[15:0]             | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      |
| 0x28   | TIM15_PSC     |        | PSC[15:0]             | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      |
| 0x2C   | TIM15_ARR     |        | ARR[15:0]             | 1                      | 1                      | 1                      | 1                      | 1                      | 1                      | 1                      | 1                      | 1                      | 1                      | 1                      | 1                      | 1                      | 1                      | 1                      | 1                      | 1                      | 1                      |
| 0x30   | TIM15_RCR     |        | REP[7:0]              | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      |
| 0x34   | TIM15_CCR1    |        | CCR1[15:0]            | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      |
| 0x38   | TIM15_CCR2    |        | CCR2[15:0]            | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      |
| 0x44   | TIM15_BDTR    |        | BKF[3:0]              | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      |
| 0x48   | TIM15_DCR     |        | DBL[4:0]              | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      | 0                      |

---

**Reset value**

- TIM15_CCMR1: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- TIM15_CCER: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- TIM15_CNT: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- TIM15_PSC: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- TIM15_ARR: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
- TIM15_RCR: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- TIM15_CCR1: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- TIM15_CCR2: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- TIM15_BDTR: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- TIM15_DCR: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Table 321. TIM15 register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM15_DMAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DMAB[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x60</td>
<td>TIM15_AF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x68</td>
<td>TIM15_TISEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
45.6 TIM16/TIM17 registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

45.6.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

Address offset: 0x000
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-12</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>UIFREMAP (UIF status bit remapping)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>9-8</td>
<td>CKD[1:0] (Clock division)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>ARPE (Auto-reload preload enable)</td>
<td></td>
</tr>
<tr>
<td>6-4</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>OPM (One pulse mode)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>URS (Update request source)</td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping

- 0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
- 1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

- 00: \( t_{DTS} = t_{CK\_INT} \)
- 01: \( t_{DTS} = 2 \times t_{CK\_INT} \)
- 10: \( t_{DTS} = 4 \times t_{CK\_INT} \)
- 11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable

- 0: TIMx_ARR register is not buffered
- 1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode

- 0: Counter is not stopped at update event
- 1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

- 0: Any of the following events generate an update interrupt or DMA request if enabled.
  These events can be:
  - Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller
- 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.
Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

- **0**: UEV enabled. The Update (UEV) event is generated by one of the following events:
  - Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller

  Buffered registers are then loaded with their preload values.

- **1**: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

- **0**: Counter disabled
- **1**: Counter enabled

*Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

### 45.6.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

Address offset: 0x04

Reset value: 0x0000

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

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

- **0**: OC1N=0 after a dead-time when MOE=0
- **1**: OC1N=1 after a dead-time when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

- **0**: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
- **1**: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

- **0**: CCx DMA request sent when CCx event occurs
- **1**: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

- **0**: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.
- **1**: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

*Note: This bit acts only on channels that have a complementary output.*

Bit 1 Reserved, must be kept at reset value.
Bit 0  **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.

*Note:* This bit acts only on channels that have a complementary output.

### 45.6.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

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

Bits 15:10 Reserved, must be kept at reset value.

Bit 9  **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8  **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7  **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5  **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1  **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0  **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled
45.6.4  TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:10</td>
<td>Reserved, must be kept at reset value.</td>
<td>rc_w0</td>
</tr>
<tr>
<td>9</td>
<td>CC1OF: Capture/Compare 1 overcapture flag</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware only when the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>corresponding channel is configured in input</td>
<td></td>
</tr>
<tr>
<td></td>
<td>capture mode. It is cleared by software by</td>
<td></td>
</tr>
<tr>
<td></td>
<td>writing it to '0'.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No overflow has been detected</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: The counter value has been captured in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TIMx_CCR1 register while CC1IF flag was</td>
<td></td>
</tr>
<tr>
<td></td>
<td>already set</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Reserved, must be kept at reset value.</td>
<td>rc_w0</td>
</tr>
<tr>
<td>7</td>
<td>BIF: Break interrupt flag</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware as soon as the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>break input goes active. It can be cleared by</td>
<td></td>
</tr>
<tr>
<td></td>
<td>software if the break input is not active.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No break event occurred</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: An active level has been detected on the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>break input</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Reserved, must be kept at reset value.</td>
<td>rc_w0</td>
</tr>
<tr>
<td>5</td>
<td>COMIF: COM interrupt flag</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware on a COM event</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(once the capture/compare control bits –CCxE,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CCxNE, OCxM– have been updated). It is cleared</td>
<td></td>
</tr>
<tr>
<td></td>
<td>by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No COM event occurred</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: COM interrupt pending</td>
<td></td>
</tr>
<tr>
<td>4:2</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CC1IF: Capture/Compare 1 interrupt flag</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware. It is cleared by</td>
<td></td>
</tr>
<tr>
<td></td>
<td>software (input capture or output compare</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mode) or by reading the TIMx_CCR1 register</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(input capture mode only).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No compare match / No input capture occurred</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: A compare match or an input capture occurred</td>
<td></td>
</tr>
</tbody>
</table>

**If channel CC1 is configured as output:** this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

**If channel CC1 is configured as input:** this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).
Bit 0  **UIF**: Update interrupt flag  
This bit is set by hardware on an update event. It is cleared by software.  
0: No update occurred.  
1: Update interrupt pending. This bit is set by hardware when the registers are updated:  
   – At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.  
   – When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

### 45.6.5  **TIMx event generation register (TIMx_EGR)** *(x = 16 to 17)*

Address offset: 0x14  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BG</td>
<td>COMG</td>
<td>CC1G</td>
<td>UG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:8  Reserved, must be kept at reset value.

Bit 7  **BG**: Break generation  
This bit is set by software in order to generate an event, it is automatically cleared by hardware.  
0: No action.  
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6  Reserved, must be kept at reset value.

Bit 5  **COMG**: Capture/Compare control update generation  
This bit can be set by software, it is automatically cleared by hardware.  
0: No action  
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

*Note: This bit acts only on channels that have a complementary output.*

Bits 4:2  Reserved, must be kept at reset value.

Bit 1  **CC1G**: Capture/Compare 1 generation  
This bit is set by software in order to generate an event, it is automatically cleared by hardware.  
0: No action.  
1: A capture/compare event is generated on channel 1:  
   **If channel CC1 is configured as output:**  
   CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.  
   **If channel CC1 is configured as input:**  
   The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0  **UG**: Update generation  
This bit can be set by software, it is automatically cleared by hardware.  
0: No action.  
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).
45.6.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Input capture mode:**

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

Bits 31:8 Reserved, must be kept at reset value.
Bits 7:4  **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample Ti1 input and the length of the digital filter applied to Ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at fDTS
- 0001: fSAMPLING=fCK_INT, N=2
- 0010: fSAMPLING=fCK_INT, N=4
- 0011: fSAMPLING=fDTS/2, N=
- 0100: fSAMPLING=fDTS/4, N=8
- 0101: fSAMPLING=fDTS/4, N=8
- 0110: fSAMPLING=fDTS/8, N=6
- 0111: fSAMPLING=fDTS/8, N=8
- 1000: fSAMPLING=fDTS/16, N=5
- 1001: fSAMPLING=fDTS/16, N=6
- 1010: fSAMPLING=fDTS/32, N=5
- 1011: fSAMPLING=fDTS/32, N=6
- 1100: fSAMPLING=fDTS/32, N=8
- 1101: fSAMPLING=fDTS/32, N=8

Bits 3:2  **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

- 00: no prescaler, capture is done each time an edge is detected on the capture input.
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0  **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, IC1 is mapped on Ti1
- Others: Reserved

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).*

45.6.7  **TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)**

(x = 16 to 17)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Output compare mode:**
Bits 31:17  Reserved, must be kept at reset value.

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

Bits 16, 6, 4  **OC1M[3:0]:** Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

All other values: Reserved

**Note:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

In **PWM mode 1 or 2**, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

Bit 3  **OC1PE:** Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

**Note:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

The **PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register).** Else the behavior is not guaranteed.

Bit 2  **OC1FE:** Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.
Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
Others: Reserved
*Note:* **CC1S bits are writable only when the channel is OFF (CC1E = ‘0’ in TIMx CCER).**

### 45.6.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20
Reset value: 0x0000

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

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity
CC1 channel configured as output:
0: OC1N active high
1: OC1N active low

CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to the description of CC1P.

*Note:* This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx BDTR register) and CC1S=“00” (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.
Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

- CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode). TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).
- CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode). TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
- CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode). TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
- CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to **Table 322** for details.
Table 322. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)

<table>
<thead>
<tr>
<th>Control bits</th>
<th>Output states(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOE bit</td>
<td>OSSi bit</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and AFIO registers.

45.6.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24
Reset value: 0x0000 0000

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

1748/2974
45.6.10 TIMx prescaler (TIMx_PSC) (x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

<table>
<thead>
<tr>
<th>PSC[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in "reset mode").

45.6.11 TIMx auto-reload register (TIMx_ARR) (x = 16 to 17)

Address offset: 0x2C

Reset value: 0xFFFF

<table>
<thead>
<tr>
<th>ARR[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the Section 45.4.1: Time-base unit on page 1676 for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.
45.6.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30
Reset value: 0x0000

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

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]: Repetition counter value**

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enabled, as well as the update interrupt generation rate, if this interrupt is enable. Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event. It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

45.6.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34
Reset value: 0x00000

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

Bits 15:0 **CCR1[15:0]: Capture/Compare 1 value**

**If channel CC1 is configured as output:**
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

**If channel CC1 is configured as input:**
CCR1 is the counter value transferred by the last input capture 1 event (IC1).
45.6.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44
Reset value: 0x0000 0000

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

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
MOE AOE BKP BKE OSSR OSSI LOCK[1:0] DTG[7:0]

As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional
0: Break input BRK in input mode
1: Break input BRK in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

*Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm
0: Break input BRK is armed
1: Break input BRK is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*
Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter
This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
- 0000: No filter, BRK acts asynchronously
- 0001: $f_{SAMPLING} = f_{CK_INT}$, $N = 2$
- 0010: $f_{SAMPLING} = f_{CK_INT}$, $N = 4$
- 0011: $f_{SAMPLING} = f_{CK_INT}$, $N = 8$
- 0100: $f_{SAMPLING} = f_{DTS}/2$, $N = 6$
- 0101: $f_{SAMPLING} = f_{DTS}/2$, $N = 8$
- 0110: $f_{SAMPLING} = f_{DTS}/4$, $N = 6$
- 0111: $f_{SAMPLING} = f_{DTS}/4$, $N = 8$
- 1000: $f_{SAMPLING} = f_{DTS}/8$, $N = 6$
- 1001: $f_{SAMPLING} = f_{DTS}/8$, $N = 8$
- 1010: $f_{SAMPLING} = f_{DTS}/16$, $N = 5$
- 1011: $f_{SAMPLING} = f_{DTS}/16$, $N = 6$
- 1100: $f_{SAMPLING} = f_{DTS}/16$, $N = 8$
- 1101: $f_{SAMPLING} = f_{DTS}/32$, $N = 5$
- 1110: $f_{SAMPLING} = f_{DTS}/32$, $N = 6$
- 1111: $f_{SAMPLING} = f_{DTS}/32$, $N = 8$

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable
This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.
- 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
- 1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)

See OC/OCN enable description for more details (Section 45.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 1746).

Bit 14 **AOE**: Automatic output enable
- 0: MOE can be set only by software
- 1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity
- 0: Break input BRK is active low
- 1: Break input BRK is active high

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable
- 0: Break inputs (BRK and CCS clock failure event) disabled
- 1: Break inputs (BRK and CCS clock failure event) enabled

*Note:* This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (Section 45.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 1746).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)
1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

**Note**: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See OC/OCN enable description for more details (Section 45.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 1746).

0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)
1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

**Note**: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

**Note**: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x t_{tdg} with t_{tdg}=1\_DTS
DTG[7:5]=10x => DT=([64+DTG[5:0]])x t_{tdg} with T_{tdg}=2x_{DTS}
DTG[7:5]=110 => DT=([32+DTG[4:0]])x t_{tdg} with T_{tdg}=8x_{DTS}
DTG[7:5]=111 => DT=([32+DTG[4:0]])x t_{tdg} with T_{tdg}=16x_{DTS}

Example if T_{DTS}=125\_ns (8\_MHz), dead-time possible values are:
0 to 15875 \_ns by 125 \_ns steps,
16 \_µs to 31750 \_ns by 250 \_ns steps,
32 \_µs to 63 \_µs by 1 \_µs steps,
64 \_µs to 126 \_µs by 2 \_µs steps

**Note**: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).
45.6.15 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

Address offset: 0x48
Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

- 00000: 1 transfer,
- 00001: 2 transfers,
- 00010: 3 transfers,
- ...
- 10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

- 00000: TIMx_CR1,
- 00001: TIMx_CR2,
- 00010: TIMx_SMCR,
- ...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

45.6.16 TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

Address offset: 0x4C
Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).
### 45.6.17 TIM16 alternate function register 1 (TIM16_AF1)

Address offset: 0x60  
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:12</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>BKCMP2P: BRK COMP2 input polarity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>BKCMP1P: BRK COMP1 input polarity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>BKINP: BRK BKIN input polarity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>BKDFBK1E: BRK dfsdm1_break[1] enable</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:12 Reserved, must be kept at reset value.**

**Bit 11 BKCMP2P: BRK COMP2 input polarity**

- This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
  - 0: COMP2 input is active low
  - 1: COMP2 input is active high

  *Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

**Bit 10 BKCMP1P: BRK COMP1 input polarity**

- This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
  - 0: COMP1 input is active low
  - 1: COMP1 input is active high

  *Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

**Bit 9 BKINP: BRK BKIN input polarity**

- This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
  - 0: BKIN input is active low
  - 1: BKIN input is active high

  *Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

**Bit 8 BKDFBK1E: BRK dfsdm1_break[1] enable**

- This bit enables the dfsdm1_break[1] for the timer’s BRK input. dfsdm1_break[1] output is ‘OREd’ with the other BRK sources.
  - 0: dfsdm1_break[1] input disabled
  - 1: dfsdm1_break[1] input enabled

  *Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

**Bits 7:3 Reserved, must be kept at reset value.**
45.6.18 TIM16 input selection register (TIM16_TISEL)

Address offset: 0x68
Reset value: 0x0000 0000

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

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

Bits 3:0 **TIMSEL[3:0]**: selects T11[0] to T11[15] input
- 0000: TIM16_CH1 input
- 0001: LSI
- 0010: LSE
- 0011: WKUP_IT
- Other: Reserved
45.6.19 TIM17 alternate function register 1 (TIM17_AF1)

Address offset: 0x60
Reset value: 0x0000 0001

| Bit 31:12 Reserved, must be kept at reset value. |
| Bit 11 **BKCMP2P**: BRK COMP2 input polarity |
| This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit. |
| 0: COMP2 input is active low |
| 1: COMP2 input is active high |
| **Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register). |

| Bit 10 **BKCMP1P**: BRK COMP1 input polarity |
| This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit. |
| 0: COMP1 input is active low |
| 1: COMP1 input is active high |
| **Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register). |

| Bit 9 **BKINP**: BRK BKIN input polarity |
| This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit. |
| 0: BKIN input is active low |
| 1: BKIN input is active high |
| **Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register). |

| Bit 8 **BKDF1BK2E**: BRK dfsdm1_break[2] enable |
| This bit enables the dfsdm1_break[2] for the timer’s BRK input. dfsdm1_break[2] output is ‘OREd’ with the other BRK sources. |
| 0: dfsdm1_break[2] input disabled |
| 1: dfsdm1_break[2] input enabled |
| **Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register). |

Bits 7:3 Reserved, must be kept at reset value.
### TIM17 input selection register (TIM17_TISEL)

**Address offset:** 0x68  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15:4</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>3:0</td>
<td>TI1SEL[3:0]: selects TI1[0] to TI1[15] input</td>
</tr>
<tr>
<td></td>
<td>0000: TIM17_CH1 input</td>
</tr>
<tr>
<td></td>
<td>0010: HSE_1MHz</td>
</tr>
<tr>
<td></td>
<td>0011: MCO1</td>
</tr>
<tr>
<td></td>
<td>Others: Reserved</td>
</tr>
</tbody>
</table>

- **Bit 2** **BKCMP2E**: BRK COMP2 enable  
  This bit enables the COMP2 for the timer’s BRK input. COMP2 output is ‘ORed’ with the other BRK sources.  
  0: COMP2 input disabled  
  1: COMP2 input enabled  
  **Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

- **Bit 1** **BKCMP1E**: BRK COMP1 enable  
  This bit enables the COMP1 for the timer’s BRK input. COMP1 output is ‘ORed’ with the other BRK sources.  
  0: COMP1 input disabled  
  1: COMP1 input enabled  
  **Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

- **Bit 0** **BKINE**: BRK BKIN input enable  
  This bit enables the BKIN alternate function input for the timer’s BRK input. BKIN input is ‘ORed’ with the other BRK sources.  
  0: BKIN input disabled  
  1: BKIN input enabled  
  **Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
### 45.6.21 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 323. TIM16/TIM17 register map and reset values**

| Offset | Register name | Offset Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | TIMx_CR1      |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | TIMx_CR2      |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | TIMx_DIER     |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | TIMx_SR       |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x14   | TIMx_EGR      |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x18   | TIMx_CCMR1    | Output Compare mode |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x1C   | TIMx_CCMR1    | Input Capture mode  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x20   | TIMx_CCER     |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x24   | TIMx_CNT      |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x28   | TIMx_PSC      |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2C   | TIMx_ARR      |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

- **Reset value**: Represents the default value of the register when the device is reset.
- **Offset**: Represents the memory address of the register.
- **Register name**: Names the register.
- **Offset Register name**: Describes the specific offsets within the register.
- **Value**: Shows the hexadecimal value of the register's configuration at a given offset.
Refer to Section 2.3 on page 128 for the register boundary addresses.
46 Basic timers (TIM6/TIM7)

46.1 TIM6/TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

The timers are completely independent, and do not share any resources.

46.2 TIM6/TIM7 main features

Basic timer (TIM6/TIM7) features include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 482. Basic timer block diagram
46.3 TIM6/TIM7 functional description

46.3.1 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:
- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 483 and Figure 484 give some examples of the counter behavior when the prescaler ratio is changed on the fly.
Figure 483. Counter timing diagram with prescaler division change from 1 to 2

Figure 484. Counter timing diagram with prescaler division change from 1 to 4
46.3.2 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 485. Counter timing diagram, internal clock divided by 1
Figure 486. Counter timing diagram, internal clock divided by 2

Figure 487. Counter timing diagram, internal clock divided by 4
Figure 488. Counter timing diagram, internal clock divided by N

Figure 489. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)
46.3.3 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register’s bit 31 (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

46.3.4 Clock source

The counter clock is provided by the Internal clock (CK_INT) source.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 491 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.
46.3.5 Debug mode

When the microcontroller enters the debug mode (Cortex®-M7 core - halted), the TIMx counter either continues to work normally or stops, depending on the DBG_TIMx_STOP configuration bit in the DBGMCU module. For more details, refer to Section 60.5.8: Microcontroller debug unit (DBGMCU).

46.4 TIM6/TIM7 registers

Refer to Section 1.2 on page 97 for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

46.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

Address offset: 0x00
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>
Bit 7 **ARPE**: Auto-reload preload enable
   - 0: TIMx_ARR register is not buffered.
   - 1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
   - 0: Counter is not stopped at update event
   - 1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source
   This bit is set and cleared by software to select the UEV event sources.
   - 0: Any of the following events generates an update interrupt or DMA request if enabled.
     These events can be:
     - Counter overflow/underflow
     - Setting the UG bit
     - Update generation through the slave mode controller
   - 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
   This bit is set and cleared by software to enable/disable UEV event generation.
   - 0: UEV enabled. The Update (UEV) event is generated by one of the following events:
     - Counter overflow/underflow
     - Setting the UG bit
     - Update generation through the slave mode controller
   Buffered registers are then loaded with their preload values.
   - 1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
   - 0: Counter disabled
   - 1: Counter enabled

   **Note**: Gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

   CEN is cleared automatically in one-pulse mode, when an update event occurs.
46.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7)

Address offset: 0x04
Reset value: 0x0000

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

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

Bits 6:4 MMS[2:0]: Master mode selection
These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: Reset - the UG bit from the TIMx_EGR register is used as a trigger output (TRGO). If reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: Enable - the Counter enable signal, CNT_EN, is used as a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).
010: Update - The update event is selected as a trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

46.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)

Address offset: 0x0C
Reset value: 0x0000

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

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

Bit 8 UDE: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 UIE: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.
46.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)
Address offset: 0x10
Reset value: 0x0000

|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
- At overflow or underflow regarding the repetition counter value and if UDIS = 0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

46.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7)
Address offset: 0x14
Reset value: 0x0000

|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

46.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)
Address offset: 0x24
Reset value: 0x0000 0000

|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|---------|

<table>
<thead>
<tr>
<th>CNT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>
**46.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7)**

Address offset: 0x28

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>UIFCPY: UIF Copy</td>
</tr>
<tr>
<td></td>
<td>This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.</td>
</tr>
<tr>
<td>30:16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15:0</td>
<td>CNT[15:0]: Counter value</td>
</tr>
</tbody>
</table>

**46.4.8 TIMx auto-reload register (TIMx_ARR)(x = 6 to 7)**

Address offset: 0x2C

Reset value: 0xFFFF

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>ARR[15:0]: Prescaler value</td>
</tr>
<tr>
<td></td>
<td>ARR is the value to be loaded into the actual auto-reload register.</td>
</tr>
<tr>
<td></td>
<td>Refer to Section 46.3.1: Time-base unit on page 1762 for more details about ARR update and behavior.</td>
</tr>
<tr>
<td></td>
<td>The counter is blocked while the auto-reload value is null.</td>
</tr>
</tbody>
</table>
46.4.9 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>TIMx_CR1</td>
<td></td>
<td>0x04</td>
<td>TIMx_CR2</td>
<td></td>
<td>0x08</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>TIMx_DIER</td>
<td></td>
<td>0x0C</td>
<td>TIMx_SR</td>
<td></td>
<td>0x10</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>TIMx_SR</td>
<td></td>
<td>0x14</td>
<td>TIMx_EGR</td>
<td></td>
<td>0x18-</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td>0x20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>TIMx_EGR</td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td>0x24</td>
<td>TIMx_CNT</td>
<td>CNT[15:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
</tr>
<tr>
<td>0x24</td>
<td>TIMx_CNT</td>
<td></td>
<td>0x28</td>
<td>TIMx_PSC</td>
<td></td>
<td>0x2C</td>
<td>TIMx_ARR</td>
<td>ARR[15:0]</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td>Reset value</td>
<td>1111111111111111</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
47 Low-power timer (LPTIM)

47.1 Introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. Also, the LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

47.2 LPTIM main features

- 16 bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: LSE, LSI, HSI or APB clock
  - External clock source over LPTIM input (working with no LP oscillator running, used by Pulse Counter application)
- 16 bit ARR autoreload register
- 16 bit compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable Digital Glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
47.3 LPTIM implementation

Table 325 describes LPTIM implementation on STM32H7A3/7B3 and STM32H7B0 devices.

<table>
<thead>
<tr>
<th>LPTIM modes/features(^{(1)})</th>
<th>LPTIM1</th>
<th>LPTIM2</th>
<th>LPTIM3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder mode</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

1. X = supported.

47.4 LPTIM functional description

47.4.1 LPTIM block diagram

Figure 492. Low-power timer block diagram (LPTIM1 and LPTIM2)
47.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

**Table 326. LPTIM input/output pins**

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM_IN1</td>
<td>Digital input</td>
<td>LPTIM Input 1 from GPIO pin on mux input 0</td>
</tr>
<tr>
<td>LPTIM_IN2</td>
<td>Digital input</td>
<td>LPTIM Input 2 from GPIO pin on mux input 0</td>
</tr>
<tr>
<td>LPTIM_ETR</td>
<td>Digital input</td>
<td>LPTIM external trigger GPIO pin</td>
</tr>
<tr>
<td>LPTIM_OUT</td>
<td>Digital output</td>
<td>LPTIM Output GPIO pin</td>
</tr>
</tbody>
</table>

**Table 327. LPTIM internal signals**

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_pclk</td>
<td>Digital input</td>
<td>LPTIM APB clock domain</td>
</tr>
<tr>
<td>lptim_ker_ck</td>
<td>Digital input</td>
<td>LPTIM kernel clock</td>
</tr>
<tr>
<td>lptim_in1_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 1</td>
</tr>
<tr>
<td>lptim_in1_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 2</td>
</tr>
<tr>
<td>lptim_in1_mux3</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 3</td>
</tr>
<tr>
<td>lptim_in2_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input 2 connected tomux input 1</td>
</tr>
<tr>
<td>lptim_in2_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input 2 connected to mux input 2</td>
</tr>
</tbody>
</table>
47.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter:

**Table 328. LPTIM1 external trigger connection**

<table>
<thead>
<tr>
<th>TRIGSEL</th>
<th>External trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_ext_trig0</td>
<td>GPIO pin as LPTIM1_ETR alternate function</td>
</tr>
<tr>
<td>lptim_ext_trig1</td>
<td>RTC_ALARMA</td>
</tr>
<tr>
<td>lptim_ext_trig2</td>
<td>RTC_ALARMB</td>
</tr>
<tr>
<td>lptim_ext_trig3</td>
<td>RTC_TAMP1_OUT</td>
</tr>
<tr>
<td>lptim_ext_trig4</td>
<td>RTC_TAMP2_OUT</td>
</tr>
<tr>
<td>lptim_ext_trig5</td>
<td>RTC_TAMP3_OUT</td>
</tr>
<tr>
<td>lptim_ext_trig6</td>
<td>COMP1_OUT</td>
</tr>
<tr>
<td>lptim_ext_trig7</td>
<td>COMP2_OUT</td>
</tr>
</tbody>
</table>

**Table 329. LPTIM2 external trigger connection**

<table>
<thead>
<tr>
<th>TRIGSEL</th>
<th>External trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_ext_trig0</td>
<td>GPIO pin as LPTIM2_ETR alternate function</td>
</tr>
<tr>
<td>lptim_ext_trig1</td>
<td>RTC_ALARMA</td>
</tr>
<tr>
<td>lptim_ext_trig2</td>
<td>RTC_ALARMB</td>
</tr>
<tr>
<td>lptim_ext_trig3</td>
<td>RTC_TAMP1_OUT</td>
</tr>
<tr>
<td>lptim_ext_trig4</td>
<td>RTC_TAMP2_OUT</td>
</tr>
<tr>
<td>lptim_ext_trig5</td>
<td>RTC_TAMP3_OUT</td>
</tr>
<tr>
<td>lptim_ext_trig6</td>
<td>COMP1_OUT</td>
</tr>
<tr>
<td>lptim_ext_trig7</td>
<td>COMP2_OUT</td>
</tr>
</tbody>
</table>
### Table 330. LPTIM3 external trigger connection

<table>
<thead>
<tr>
<th>TRIGSEL</th>
<th>External trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_ext_trig0</td>
<td>LPTIM2_OUT</td>
</tr>
<tr>
<td>lptim_ext_trig1</td>
<td>Not connected</td>
</tr>
<tr>
<td>lptim_ext_trig2</td>
<td>Not connected</td>
</tr>
<tr>
<td>lptim_ext_trig3</td>
<td>Not connected</td>
</tr>
<tr>
<td>lptim_ext_trig4</td>
<td>SAI1_FS_A</td>
</tr>
<tr>
<td>lptim_ext_trig5</td>
<td>SAI1_FS_B</td>
</tr>
<tr>
<td>lptim_ext_trig6</td>
<td>DFSDM2_BRK</td>
</tr>
<tr>
<td>lptim_ext_trig7</td>
<td>Not connected</td>
</tr>
</tbody>
</table>

### Table 331. LPTIM1 input 1 connection

<table>
<thead>
<tr>
<th>lptim_in1_mux</th>
<th>LPTIM1 input 1 connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_in1_mux0</td>
<td>GPIO pin as LPTIM1_IN1 alternate function</td>
</tr>
<tr>
<td>lptim_in1_mux1</td>
<td>COMP1_OUT</td>
</tr>
<tr>
<td>lptim_in1_mux2</td>
<td>Not connected</td>
</tr>
<tr>
<td>lptim_in1_mux3</td>
<td>Not connected</td>
</tr>
</tbody>
</table>

### Table 332. LPTIM1 input 2 connection

<table>
<thead>
<tr>
<th>lptim_in2_mux</th>
<th>LPTIM1 input 2 connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_in2_mux0</td>
<td>GPIO pin as LPTIM1_IN2 alternate function</td>
</tr>
<tr>
<td>lptim_in2_mux1</td>
<td>COMP2_OUT</td>
</tr>
<tr>
<td>lptim_in2_mux2</td>
<td>Not connected</td>
</tr>
<tr>
<td>lptim_in2_mux3</td>
<td>Not connected</td>
</tr>
</tbody>
</table>

### Table 333. LPTIM2 input 1 connection

<table>
<thead>
<tr>
<th>lptim_in1_mux</th>
<th>LPTIM2 input 1 connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_in1_mux0</td>
<td>GPIO pin as LPTIM2_IN1 alternate function</td>
</tr>
<tr>
<td>lptim_in1_mux1</td>
<td>COMP1_OUT</td>
</tr>
<tr>
<td>lptim_in1_mux2</td>
<td>COMP2_OUT</td>
</tr>
<tr>
<td>lptim_in1_mux3</td>
<td>COMP1_OUT OR COMP2_OUT</td>
</tr>
</tbody>
</table>

### Table 334. LPTIM2 input 2 connection

<table>
<thead>
<tr>
<th>lptim_in2_mux</th>
<th>LPTIM2 input 2 connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_in2_mux0</td>
<td>GPIO pin as LPTIM2_IN2 alternate function</td>
</tr>
<tr>
<td>lptim_in2_mux1</td>
<td>COMP2_OUT</td>
</tr>
</tbody>
</table>
47.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be chosen among APB, LSI, LSE or HSI sources through the Reset and Clock controller (RCC). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM may run in one of these two possible configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM either from APB or any other embedded oscillator including LSE, LSI and HSI.
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or Pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM will use an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal should also be provided (first configuration). In this case, the internal clock signal frequency should be at least four times higher than the external clock signal frequency.

47.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source should first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

The digital filters are divided into two groups:

- The first group of digital filters protects the LPTIM external inputs. The digital filters sensitivity is controlled by the CKFLT bits.
- The second group of digital filters protects the LPTIM internal trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that should be detected on one of the LPTIM inputs to consider a signal level change as a valid transition. Figure 494 shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.
Figure 494. Glitch filter timing diagram

Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT and TRGFLT bits to ‘0’. In that case, an external analog filter may be used to protect the LPTIM external inputs against glitches.

47.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] 3-bit field. The table below lists all the possible division ratios:

<table>
<thead>
<tr>
<th>programming</th>
<th>dividing factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>/1</td>
</tr>
<tr>
<td>001</td>
<td>/2</td>
</tr>
<tr>
<td>010</td>
<td>/4</td>
</tr>
<tr>
<td>011</td>
<td>/8</td>
</tr>
<tr>
<td>100</td>
<td>/16</td>
</tr>
<tr>
<td>101</td>
<td>/32</td>
</tr>
<tr>
<td>110</td>
<td>/64</td>
</tr>
<tr>
<td>111</td>
<td>/128</td>
</tr>
</tbody>
</table>

47.4.7 Trigger multiplexer

The LPTIM counter may be started either by software or after the detection of an active edge on one of the 8 trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals ‘00’, The LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than ‘00’, TRIGSEL[2:0] is used to select which of the 8 trigger inputs is used to start the counter.
The external triggers are considered asynchronous signals for the LPTIM. So after a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it will be ignored (unless timeout function is enabled).

*Note:* The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled will be discarded by hardware.

### 47.4.8 Operating mode

The LPTIM features two operating modes:

- The Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when reaching the ARR value.

**One-shot mode**

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event will re-start the timer. Any trigger event occurring after the counter starts and before the counter reaches ARR will be discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the counter register has stopped (contains zero value), will start the counter for a new one-shot counting cycle as shown in *Figure 495*.

![Figure 495. LPTIM output waveform, single counting mode configuration](image)

**- Set-once mode activated:**

It should be noted that when the WAVE bit-field in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 496*.
In case of software start (TRIGEN[1:0] = ‘00’), the SNGSTRT setting will start the counter for one-shot counting.

**Continuous mode**

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set will start the counter for continuous counting. Any subsequent external trigger event will be discarded as shown in Figure 497.

In case of software start (TRIGEN[1:0] = ‘00’), setting CNTSTRT will start the counter for continuous counting.

SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (The ENABLE bit is set to ‘1’). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT will switch the LPTIM to the One-shot mode. The counter (if active) will stop as soon as it reaches ARR.

If the One-shot mode was previously selected, setting CNTSTRT will switch the LPTIM to the Continuous mode. The counter (if active) will restart as soon as it reaches ARR.
47.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event will start the timer, any successive trigger event will reset the counter and the timer will restart.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

47.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CMP (compare register), are used to generate several different waveforms on LPTIM output.

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CMP. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT registers.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset.
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require that the LPTIM_ARR register value be strictly greater than the LPTIM_CMP register value.

The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to ‘0’ forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to ‘1’ forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value will change immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated. Figure 498 below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL bit.
47.4.11 Register update

The LPTIM_ARR register and LPTIM_CMP register are updated immediately after the APB bus write operation, or at the end of the current period if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR and the LPTIM_CMP registers are updated:
- When the PRELOAD bit is reset to ‘0’, the LPTIM_ARR and the LPTIM_CMP registers are immediately updated after any write access.
- When the PRELOAD bit is set to ‘1’, the LPTIM_ARR and the LPTIM_CMP registers are updated at the end of the current period, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag and the CMPOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM.ARR register and the LPTIM_CMP register.

After a write to the LPTIM_ARR register or the LPTIM_CMP register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag or the CMPOK flag be set, will lead to unpredictable results.
47.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM Input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source will be used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- **CKSEL = 0:** the LPTIM is clocked by an internal clock source
  - **COUNTMODE = 0**
    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - **COUNTMODE = 1**
    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.
    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal should never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).

- **CKSEL = 1:** the LPTIM is clocked by an external clock source
  **COUNTMODE** value is don’t care.
  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external Input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.
  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.
  Since the signal injected on the LPTIM external Input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

47.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock is needed before the LPTIM is actually enabled.

The LPTIM_CFRG and LPTIM_IER registers must be modified only when the LPTIM is disabled.
47.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register to zero, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bit-field to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic will elapse before the reset is taken into account. This will make the LPTIM counter count few extra pluses between the time when the reset is trigger and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.

- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register will reset its content to zero. Asynchronous reset should be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset should be applied only when there is enough insurance that no toggle will occur on the LPTIM Input1.

It should be noted that to read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. So developer should make sure that these two mechanisms are used exclusively.

47.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two Down and Up flags in the LPTIM_ISR register. Also, an interrupt can be generated for both direction change events if enabled through the DOWNIE bit.
To activate the Encoder mode the ENC bit has to be set to ‘1’. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder’s position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

<table>
<thead>
<tr>
<th>Active edge</th>
<th>Level on opposite signal (Input1 for Input2, Input2 for Input1)</th>
<th>Input1 signal</th>
<th>Input2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rising</td>
<td>Falling</td>
<td>Rising</td>
</tr>
<tr>
<td>Rising Edge</td>
<td>High</td>
<td>Down</td>
<td>No count</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>No count</td>
</tr>
<tr>
<td>Falling Edge</td>
<td>High</td>
<td>No count</td>
<td>Up</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>No count</td>
<td>Down</td>
</tr>
<tr>
<td>Both Edges</td>
<td>High</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>Down</td>
</tr>
</tbody>
</table>

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

**Caution:** In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to ‘0’. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be ‘000’).
47.4.16 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the DBG_LPTIM_STOP configuration bit in the DBG module.

47.5 LPTIM low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. LPTIM interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The LPTIM peripheral is active when it is clocked by LSE or LSI. LPTIM interrupts cause the device to exit Stop mode</td>
</tr>
<tr>
<td>Standby</td>
<td>The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>
47.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_IER register:

- Compare match
- Auto-reload match (whatever the direction if encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (encoder mode), programmable (up / down / both).

*Note:* If any bit in the LPTIM_IER register (Interrupt Enable Register) is set after that its corresponding flag in the LPTIM_ISR register (Status Register) is set, the interrupt is not asserted.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compare match</td>
<td>Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the compare register (LPTIM_CMP).</td>
</tr>
<tr>
<td>Auto-reload match</td>
<td>Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the Auto-reload register (LPTIM_ARR).</td>
</tr>
<tr>
<td>External trigger event</td>
<td>Interrupt flag is raised when an external trigger event is detected</td>
</tr>
<tr>
<td>Auto-reload register</td>
<td>Interrupt flag is raised when the write operation to the LPTIM_ARR register is complete.</td>
</tr>
<tr>
<td>update OK</td>
<td></td>
</tr>
<tr>
<td>Compare register</td>
<td>Interrupt flag is raised when the write operation to the LPTIM_CMP register is complete.</td>
</tr>
<tr>
<td>update OK</td>
<td></td>
</tr>
<tr>
<td>Direction change</td>
<td>Used in Encoder mode. Two interrupt flags are embedded to signal direction change:</td>
</tr>
<tr>
<td></td>
<td>– UP flag signals up-counting direction change</td>
</tr>
<tr>
<td></td>
<td>– DOWN flag signals down-counting direction change</td>
</tr>
</tbody>
</table>

47.7 LPTIM registers

The peripheral registers can only be accessed by words (32-bit).
47.7.1 LPTIM interrupt and status register (LPTIM_ISR)

Address offset: 0x000
Reset value: 0x0000 0000

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 DOWN: Counter direction change up to down
In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 47.3: LPTIM implementation.

Bit 5 UP: Counter direction change down to up
In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 47.3: LPTIM implementation.

Bit 4 ARROK: Autoreload register update OK
ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 CMPOK: Compare register update OK
CMPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_CMP register has been successfully completed.

Bit 2 EXTTRIG: External trigger edge event
EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 ARR: Autoreload match
ARR is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARR flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 CMP: Compare match
The CMP bit is set by hardware to inform application that LPTIM_CNT register value reached the LPTIM_CMP register's value.
47.7.2  LPTIM interrupt clear register (LPTIM_ICR)

Address offset: 0x004
Reset value: 0x0000 0000

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

Bits 31:9  Reserved, must be kept at reset value.

Bits 8:7  Reserved, must be kept at reset value.

Bit 6  DOWNCF: Direction change to down clear flag
Writing 1 to this bit clears the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 47.3: LPTIM implementation.

Bit 5  UPCF: Direction change to UP clear flag
Writing 1 to this bit clears the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 47.3: LPTIM implementation.

Bit 4  ARROKCF: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3  CMPOKCF: Compare register update OK clear flag
Writing 1 to this bit clears the CMPOK flag in the LPTIM_ISR register

Bit 2  EXTTRIGCF: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1  ARRMCF: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0  CMPMCF: Compare match clear flag
Writing 1 to this bit clears the CMP flag in the LPTIM_ISR register

47.7.3  LPTIM interrupt enable register (LPTIM_IER)

Address offset: 0x008
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>DOWN E</th>
<th>UPIE</th>
<th>ARRO KIE</th>
<th>CMPO KIE</th>
<th>EXT TRIGIE</th>
<th>ARRM IE</th>
<th>CMPM IE</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Caution: The LPTIM_IER register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’)

47.7.4 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C
Reset value: 0x0000 0000
Bit 24 **ENC**: Encoder mode enable
The ENC bit controls the Encoder mode
0: Encoder mode disabled
1: Encoder mode enabled
*Note*: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 47.3: LPTIM implementation.

Bit 23 **COUNTMODE**: Counter mode enabled
The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:
0: The counter is incremented following each internal clock pulse
1: The counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode
The PRELOAD bit controls the LPTIM_ARR and the LPTIM_CMP registers update modality
0: Registers are updated after each APB bus write access
1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity
The WAVPOL bit controls the output polarity
0: The LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CMP registers
1: The LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CMP registers

Bit 20 **WAVE**: Waveform shape
The WAVE bit controls the output shape
0: Deactivate Set-once mode, PWM or One Pulse waveform depending on how the timer was started, CNTSTRT for PWM or SNGSTRT for One Pulse waveform.
1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable
The TIMOUT bit controls the Timeout feature
0: A trigger event arriving when the timer is already started will be ignored
1: A trigger event arriving when the timer is already started will reset and restart the counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity
The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:
00: software trigger (counting start is initiated by software)
01: rising edge is the active edge
10: falling edge is the active edge
11: both edges are active edges

Bit 16 **Reserved**, must be kept at reset value.
Bits 15:13 **TRIGSEL[2:0]**: Trigger selector
The TRIGSEL bits select the trigger source that will serve as a trigger event for the LPTIM among the below 8 available sources:
- 000: `lptim_ext_trig0`
- 001: `lptim_ext_trig1`
- 010: `lptim_ext_trig2`
- 011: `lptim_ext_trig3`
- 100: `lptim_ext_trig4`
- 101: `lptim_ext_trig5`
- 110: `lptim_ext_trig6`
- 111: `lptim_ext_trig7`
See Section 47.4.3: LPTIM input and trigger mapping for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]**: Clock prescaler
The PRESC bits configure the prescaler division factor. It can be one among the following division factors:
- 000: /1
- 001: /2
- 010: /4
- 011: /8
- 100: /16
- 101: /32
- 110: /64
- 111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]**: Configurable digital filter for trigger
The TRGFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature:
- 00: any trigger active level change is considered as a valid trigger
- 01: trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
- 10: trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
- 11: trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.
Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock  
The CKFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature  
00: any external clock signal level change is considered as a valid transition  
01: external clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.  
10: external clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.  
11: external clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.  

Bits 2:1 **CKPOL[1:0]:** Clock Polarity  
If LPTIM is clocked by an external clock source:  
When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:  
00: the rising edge is the active edge used for counting.  
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 1 is active.  
01: the falling edge is the active edge used for counting  
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 2 is active.  
10: both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.  
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 3 is active.  
11: not allowed  
Refer to **Section 47.4.15: Encoder mode** for more details about Encoder mode sub-modes.  

Bit 0 **CKSEL:** Clock selector  
The CKSEL bit selects which clock source the LPTIM will use:  
0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)  
1: LPTIM is clocked by an external clock source through the LPTIM external Input1  

Caution: The LPTIM_CFG register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’).

**47.7.5 LPTIM control register (LPTIM_CR)**

Address offset: 0x010  
Reset value: 0x0000 0000

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rs</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE**: Reset after read enable
This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register will asynchronously reset LPTIM_CNT register content.

Bit 3 **COUNTRST**: Counter reset
This bit is set by software and cleared by hardware. When set to '1' this bit will trigger a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock may be different from APB clock).

**Caution**: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software should consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT**: Timer start in Continuous mode
This bit is set by software and cleared by hardware.
In case of software start (TRIGEN[1:0] = '00'), setting this bit starts the LPTIM in Continuous mode.
If the software start is disabled (TRIGEN[1:0] different than '00'), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.
If this bit is set when a single pulse mode counting is ongoing, then the timer will not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.
This bit can be set only when the LPTIM is enabled. It will be automatically reset by hardware.

Bit 1 **SNGSTRT**: LPTIM start in Single mode
This bit is set by software and cleared by hardware.
In case of software start (TRIGEN[1:0] = '00'), setting this bit starts the LPTIM in single pulse mode.
If the software start is disabled (TRIGEN[1:0] different than '00'), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.
If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM will stop at the following match between LPTIM_ARR and LPTIM_CNT registers.
This bit can only be set when the LPTIM is enabled. It will be automatically reset by hardware.

Bit 0 **ENABLE**: LPTIM enable
The ENABLE bit is set and cleared by software.
0: LPTIM is disabled
1: LPTIM is enabled

### 47.7.6 LPTIM compare register (LPTIM_CMP)
Address offset: 0x014
Reset value: 0x0000 0000

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

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

**CMP[15:0]**

1796/2974 RM0455 Rev 3
Caution: The LPTIM_CMP register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

47.7.7 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018
Reset value: 0x0000 0001

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

ARR[15:0]

rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 ARR[15:0]: Auto reload value
ARR is the autoreload value for the LPTIM.
This value must be strictly greater than the CMP[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

47.7.8 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C
Reset value: 0x0000 0000

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

CNT[15:0]

r r r r r r r r r r r r r r r r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 CNT[15:0]: Counter value
When the LPTIM is running with an asynchronous clock, reading the LPTIM_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two returned values are identical.
It should be noted that for a reliable LPTIM_CNT register read access, two consecutive read accesses must be performed and compared. A read access can be considered reliable when the values of the two consecutive read accesses are equal.
47.7.9 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:6</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 5:4</td>
<td>IN2SEL[1:0]: LPTIM input 2 selection</td>
</tr>
<tr>
<td></td>
<td>The IN2SEL bits control the LPTIM Input 2 multiplexer, which connect LPTIM Input 2 to one of the available inputs.</td>
</tr>
<tr>
<td></td>
<td>00: lptim_in2_mux0</td>
</tr>
<tr>
<td></td>
<td>01: lptim_in2_mux1</td>
</tr>
<tr>
<td></td>
<td>10: lptim_in2_mux2</td>
</tr>
<tr>
<td></td>
<td>11: lptim_in2_mux3</td>
</tr>
<tr>
<td>Note:</td>
<td>If the LPTIM does not support encoder mode feature, these bits are reserved. Please refer to Section 47.3: LPTIM implementation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 3:2</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 1:0</td>
<td>IN1SEL[1:0]: LPTIM input 1 selection</td>
</tr>
<tr>
<td></td>
<td>The IN1SEL bits control the LPTIM Input 1 multiplexer, which connects LPTIM Input 1 to one of the available inputs.</td>
</tr>
<tr>
<td></td>
<td>00: lptim_in1_mux0</td>
</tr>
<tr>
<td></td>
<td>01: lptim_in1_mux1</td>
</tr>
<tr>
<td></td>
<td>10: lptim_in1_mux2</td>
</tr>
<tr>
<td></td>
<td>11: lptim_in1_mux3</td>
</tr>
<tr>
<td>Note:</td>
<td>If LPTIM does not implement external input clock, these bits are reserved. Please refer to Section 47.3: LPTIM implementation.</td>
</tr>
</tbody>
</table>

Caution: The LPTIM_CFGR2 register must only be modified when the LPTIM is disabled (ENABLE bit reset to '0').
## 47.7.10 LPTIM register map

The following table summarizes the LPTIM registers.

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | LPTIM_ISR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |                  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x004  | LPTIM_ICR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |                  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x008  | LPTIM_IER     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |                  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x00C  | LPTIM_CFGR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |                  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x010  | LPTIM_CR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |                  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x014  | LPTIM_CMP     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |                  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x018  | LPTIM_ARR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |                  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0x01C  | LPTIM_CNT     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |                  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x024  | LPTIM_CFGR2   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |                  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. If LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 47.3: LPTIM implementation.
Refer to Section 2.3 on page 128 for the register boundary addresses.
48 System window watchdog (WWDG)

48.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit becomes cleared. A reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter has reached the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications which require the watchdog to react within an accurate timing window.

48.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see Figure 501)
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40.

48.3 WWDG functional description

If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent . This operation must occur only when the counter value is lower than the window register value and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to Figure 500 and to Section 48.3.2: WWDG internal signals for the WWDG block diagram.
48.3.1 WWDG block diagram

![Figure 500. Watchdog block diagram](image)

48.3.2 WWDG internal signals

Table 340 gives the list of WWDG internal signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pclk</td>
<td>Digital input</td>
<td>APB bus clock</td>
</tr>
<tr>
<td>wwdg_out_rst</td>
<td>Digital output</td>
<td>WWDG reset signal output</td>
</tr>
<tr>
<td>wwdg_it</td>
<td>Digital output</td>
<td>WWDG early interrupt output</td>
</tr>
</tbody>
</table>

48.3.3 Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

48.3.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG_CR register (see Figure 501). The WWDG configuration register (WWDG_CFR) contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than the window register value and greater than 0x3F. Figure 501 describes the window watchdog process.
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

48.3.5 How to program the watchdog timeout

Use the formula in Figure 501 to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

The formula to calculate the timeout value is given by:

\[ t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \] (ms)

where:
- \( t_{WWDG} \): WWDG timeout
- \( t_{PCLK} \): APB clock period measured in ms
- 4096: value corresponding to internal divider

Figure 501. Window watchdog timing diagram
As an example, let's assume APB frequency is equal to 48 MHz, WDGTB[2:0] is set to 3 and T[5:0] is set to 63:

\[ t_{\text{WWDG}} = \left(\frac{1}{48000}\right) \times 4096 \times 2^3 \times (63 + 1) = 43.69\text{ms} \]

Refer to the datasheet for the minimum and maximum values of the \( t_{\text{WWDG}} \).

### 48.3.6 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details refer to [Section 64: Debug infrastructure](#).

### 48.4 WWDG interrupts

The early wakeup interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by setting the EWI bit in the WWDG_CFR register. When the down-counter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case, the corresponding interrupt service routine (ISR) has to reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

**Note:** *When the EWI interrupt cannot be served, e.g. due to a system lock in a higher priority task, the WWDG reset is eventually generated.*

### 48.5 WWDG registers

Refer to [Section 1.2 on page 97](#) for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

#### 48.5.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

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

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

| WDGA | T[6:0] | rs | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

1804/2974  RM0455 Rev 3
48.5.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004
Reset value: 0x0000 007F

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

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 WDGTB[2:0]: Timer base
The timebase of the prescaler can be modified as follows:
000: CK Counter Clock (PCLK div 4096) div 1
001: CK Counter Clock (PCLK div 4096) div 2
010: CK Counter Clock (PCLK div 4096) div 4
011: CK Counter Clock (PCLK div 4096) div 8
100: CK Counter Clock (PCLK div 4096) div 16
101: CK Counter Clock (PCLK div 4096) div 32
110: CK Counter Clock (PCLK div 4096) div 64
111: CK Counter Clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 EWI: Early wakeup interrupt
When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 W[6:0]: 7-bit window value
These bits contain the window value to be compared with the down-counter.
48.5.3 WWDG status register (WWDG_SR)

Address offset: 0x008
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>0</td>
<td>EWIF: Early wakeup interrupt flag</td>
</tr>
</tbody>
</table>

   This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing '0'. Writing '1' has no effect. This bit is also set if the interrupt is not enabled.

48.5.4 WWDG register map

The following table gives the WWDG register map and reset values.

**Table 341. WWDG register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Bits 31:0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>WWDG_CR</td>
<td>T[6:0]</td>
<td>WDGA</td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>0 1 1 1 1 1 1</td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>WWDG_CFR</td>
<td>WDGTB[2:0]</td>
<td>EWIF</td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>0 0 0 0 0 0 0</td>
<td>1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0x008</td>
<td>WWDG_SR</td>
<td>EWI</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>0 0 0 0 0 0 0 1</td>
<td>1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

Refer to *Section 2.3 on page 128* for the register boundary addresses.
49 Independent watchdog (IWDG)

49.1 Introduction

The devices feature an embedded watchdog peripheral that offers a combination of high safety level, timing accuracy and flexibility of use. The Independent watchdog peripheral detects and solves malfunctions due to software failure, and triggers system reset when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. For further information on the window watchdog, refer to Section 48 on page 1801.

49.2 IWDG main features

- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

49.3 IWDG functional description

49.3.1 IWDG block diagram

*Figure 502* shows the functional blocks of the independent watchdog module.
1. The register interface is located in the VDD voltage domain. The watchdog function is located in the VDD voltage domain, still functional in Stop and Standby modes.

When the independent watchdog is started by writing the value 0x0000 CCCC in the IWDG key register (IWDG_KR), the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0x0000 AAAA is written in the IWDG key register (IWDG_KR), the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

Once running, the IWDG cannot be stopped.

49.3.2 IWDG internal signals

Table 342 gives the list of IWDG internal signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsi_ck</td>
<td>Digital input</td>
<td>LSI clock</td>
</tr>
<tr>
<td>iwdg1_out_rst</td>
<td>Digital output</td>
<td>IWDG1 reset signal output</td>
</tr>
</tbody>
</table>

49.3.3 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the IWDG window register (IWDG_WINR).

If the reload operation is performed while the counter is greater than the value stored in the IWDG window register (IWDG_WINR), then a reset is provided.

The default value of the IWDG window register (IWDG_WINR) is 0x0000 0FFF, so if it is not updated, the window option is disabled.

As soon as the window value is changed, a reload operation is performed in order to reset the downcounter to the IWDG reload register (IWDG_RLR) value and ease the cycle number calculation to generate the next reload.
Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the IWDG key register (IWDG_KR).
2. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
3. Write the IWDG prescaler by programming IWDG prescaler register (IWDG_PR) from 0 to 7.
4. Write the IWDG reload register (IWDG_RLR).
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Write to the IWDG window register (IWDG_WINR). This automatically refreshes the counter value in the IWDG reload register (IWDG_RLR).

Note: Writing the window value allows the counter value to be refreshed by the RLR when IWDG status register (IWDG_SR) is set to 0x0000 0000.

Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the IWDG key register (IWDG_KR).
2. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
3. Write the prescaler by programming the IWDG prescaler register (IWDG_PR) from 0 to 7.
4. Write the IWDG reload register (IWDG_RLR).
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Refresh the counter value with IWDG_RLR (IWDG_KR = 0x0000 AAAA).

49.3.4 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and generates a reset unless the IWDG key register (IWDG_KR) is written by the software before the counter reaches end of count or if the downcounter is reloaded inside the window.

49.3.5 Low-power freeze

Depending on the IWDG_STOP and IWDG_STBYIWDG_FZ_STOP and IWDG_FZ_STBY options configuration, the IWDG can continue counting or not during the Stop mode and the Standby mode, respectively. If the IWDG is kept running during Stop or Standby modes, it can wake up the device from this mode. Refer to Section 4.4: FLASH option bytes.

49.3.6 Register access protection

Write access to IWDG prescaler register (IWDG_PR), IWDG reload register (IWDG_RLR) and IWDG window register (IWDG_WINR) is protected. To modify them, the user must first write the code 0x0000 5555 in the IWDG key register (IWDG_KR). A write access to this register with a different value breaks the sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or of the downcounter reload value or of the window value is ongoing.
49.3.7  **Debug mode**

When the device enters Debug mode (core halted), the IWDG counter either continues to work normally or stops, depending on the configuration of the corresponding bit in DBGMCU freeze register.
49.4 IWDG registers

Refer to Section 1.2 on page 97 for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

49.4.1 IWDG key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by Standby mode)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15:0</td>
<td>KEY[15:0]: Key value (write only, read 0x0000)</td>
</tr>
<tr>
<td></td>
<td>These bits must be written by software at regular intervals with the key value 0xAAAA, otherwise the watchdog generates a reset when the counter reaches 0.</td>
</tr>
<tr>
<td></td>
<td>Writing the key value 0x5555 to enable access to the IWDG_PR, IWDG_RLR and IWDG_WINR registers (see Section 49.3.6: Register access protection)</td>
</tr>
<tr>
<td></td>
<td>Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)</td>
</tr>
</tbody>
</table>
49.4.2  IWDG prescaler register (IWDG_PR)

Address offset: 0x04

Reset value:

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

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

Bits 2:0  PR[2:0]: Prescaler divider

These bits are write access protected see Section 49.3.6: Register access protection. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the IWDG status register (IWDG_SR) must be reset in order to be able to change the prescaler divider.

000: divider /4
001: divider /8
010: divider /16
011: divider /32
100: divider /64
101: divider /128
110: divider /256
111: divider /256

Note: Reading this register returns the prescaler value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.
49.4.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08
Reset value: 0x0000 0FFF (reset by Standby mode)

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

Bits 31:12  Reserved, must be kept at reset value.

Bits 11:0  RL[11:0]: Watchdog counter reload value

These bits are write access protected see Register access protection. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the IWDG key register (IWDG_KR). The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to the datasheet for the timeout information.

The RVU bit in the IWDG status register (IWDG_SR) must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on it. For this reason the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.
### 49.4.4 IWDG status register (IWDG_SR)

Address offset: 0x0C
Reset value: 0x0000 0000 (not reset by Standby mode)

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

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

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

Bit 2 **WVU**: Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the $V_{DD}$ voltage domain (takes up to five LSI/Prescaler clock cycles).

Window value can be updated only when WVU bit is reset.

Bit 1 **RVU**: Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the $V_{DD}$ voltage domain (takes up to five LSI/Prescaler clock cycles).

Reload value can be updated only when RVU bit is reset.

Bit 0 **PVU**: Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the $V_{DD}$ voltage domain (takes up to five LSI/Prescaler clock cycles).

Prescaler value can be updated only when PVU bit is reset.

**Note:** If several reload, prescaler, or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, and to wait until WVU bit is reset before changing the window value. However, after updating the prescaler and/or the reload/window value it is not necessary to wait until RVU or PVU or WVU is reset before continuing code execution except in case of low-power mode entry.
49.4.5 IWDG window register (IWDG_WINR)

Address offset: 0x10
Reset value: 0x0000 0FFF (reset by Standby mode)

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]**: Watchdog counter window value

These bits are write access protected, see Section 49.3.6, they contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x0

The WVU bit in the **IWDG status register (IWDG_SR)** must be reset in order to be able to change the reload value.

**Note**: Reading this register returns the reload value from the V_DD voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the **IWDG status register (IWDG_SR)** is reset.
### 49.4.6 IWDG register map

The following table gives the IWDG register map and reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>IWDG_KR</td>
<td>0x04</td>
<td>IWDG_PR</td>
<td>0x08</td>
<td>IWDG_RLR</td>
</tr>
<tr>
<td>0x0C</td>
<td>IWDG_SR</td>
<td>0x10</td>
<td>IWDG_WINR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Table 343. IWDG register map and reset values

<table>
<thead>
<tr>
<th>Offset (hex)</th>
<th>Register name</th>
<th>Offset (binary)</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>IWDG_KR</td>
<td>0x00</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x04</td>
<td>IWDG_PR</td>
<td>0x04</td>
<td>PR[2:0]</td>
</tr>
<tr>
<td>0x08</td>
<td>IWDG_RLR</td>
<td>0x08</td>
<td>RL[11:0]</td>
</tr>
<tr>
<td>0x0C</td>
<td>IWDG_SR</td>
<td>0x0C</td>
<td>RV[11:0]</td>
</tr>
<tr>
<td>0x10</td>
<td>IWDG_WINR</td>
<td>0x10</td>
<td>WIN[11:0]</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
50  Real-time clock (RTC)

50.1  Introduction

The RTC provides an automatic wakeup to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

The RTC is functional in \( V_{\text{BAT}} \) mode.

50.2  RTC main features

The RTC supports the following features (see Figure 503: RTC block diagram):

- Calendar with subsecond, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to \( V_{\text{BAT}} \) mode.
- 17-bit auto-reload wakeup timer (WUT) for periodic events with programmable resolution and period.

The RTC is supplied through a switch that takes power either from the \( V_{\text{DD}} \) supply when present or from the \( V_{\text{BAT}} \) pin.

The RTC clock sources can be:

- A 32.768 kHz external crystal (LSE)
- An external resonator or oscillator (LSE)
- The internal low power RC oscillator (LSI, with typical frequency of 32 kHz)
- The high-speed external clock (HSE), divided by a prescaler in the RCC.

The RTC is functional in \( V_{\text{BAT}} \) mode and in all low-power modes when it is clocked by the LSE. When clocked by the LSI, the RTC is not functional in \( V_{\text{BAT}} \) mode, but is functional in all low-power modes.

All RTC events (Alarm, WakeUp Timer, Timestamp) can generate an interrupt and wakeup the device from the low-power modes.
50.3 RTC functional description

50.3.1 RTC block diagram

Figure 503. RTC block diagram
### 50.3.2 RTC pins and internal signals

#### Table 344. RTC input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_TS</td>
<td>Input</td>
<td>RTC timestamp input</td>
</tr>
<tr>
<td>RTC_REFIN</td>
<td>Input</td>
<td>RTC 50 or 60 Hz reference clock input</td>
</tr>
<tr>
<td>RTC_OUT1</td>
<td>Output</td>
<td>RTC output 1</td>
</tr>
<tr>
<td>RTC_OUT2</td>
<td>Output</td>
<td>RTC output 2</td>
</tr>
</tbody>
</table>

- **RTC_OUT1** and **RTC_OUT2** which selects one of the following two outputs:
  - **CALIB**: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
  - **TAMPALRM**: This output is the OR between TAMP and ALARM outputs.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wakeup outputs. TAMP is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

#### Table 345. RTC internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtc_ker_ck</td>
<td>Input</td>
<td>RTC kernel clock, also named RTCCLK in this document</td>
</tr>
<tr>
<td>rtc_pclk</td>
<td>Input</td>
<td>RTC APB clock</td>
</tr>
<tr>
<td>rtc_it</td>
<td>Input</td>
<td>RTC internal timestamp event</td>
</tr>
<tr>
<td>rtc_tamp_evt</td>
<td>Input</td>
<td>Tamper event (internal or external) detected in TAMP peripheral</td>
</tr>
<tr>
<td>rtc_it</td>
<td>Output</td>
<td>RTC interrupts (refer to Section 50.5: RTC interrupts for details)</td>
</tr>
<tr>
<td>rtc_alra_trg</td>
<td>Output</td>
<td>RTC alarm A event detection trigger</td>
</tr>
<tr>
<td>rtc_alrb_trg</td>
<td>Output</td>
<td>RTC alarm B event detection trigger</td>
</tr>
<tr>
<td>rtc_wut_trg</td>
<td>Output</td>
<td>RTC wakeup timer event detection trigger</td>
</tr>
<tr>
<td>rtc_calovf</td>
<td>Output</td>
<td>RTC calendar overflow</td>
</tr>
</tbody>
</table>

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or \( V_{\text{BAT}} \) when the selected clock is not LSE. Refer to Section 50.4: RTC low-power modes for more details.
The triggers outputs can be used as triggers for other peripherals.

### 50.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the Battery Backup Domain (V\textsubscript{BAT}) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

Both RTC and TAMP peripherals provide functions on these I/Os (refer to Section 51: Tamper and backup registers (TAMP)).

RTC\_OUT1, RTC\_TS, TAMP\_IN1, TAMP\_OUT2 and TAMP\_OUT3 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes and in V\textsubscript{BAT} mode.

The output mechanism follows the priority order shown in Table 347.

![Table 347. RTC pin PC13 configuration(1)](image-url)

<table>
<thead>
<tr>
<th>PC13 Pin function</th>
<th>OSEL[1:0] (ALARM output enable)</th>
<th>TAMPOE (TAMPER output enable)</th>
<th>COE (CALIB output enable)</th>
<th>OUT2EN</th>
<th>TAMPALRM_TYPE</th>
<th>TAMPE_TAMPA2AM &amp; OUT3_RMP</th>
<th>TAMPE_IN1 input enable</th>
<th>TSE (RTC_TS input enable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMPALRM output Push-Pull</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>1</td>
<td>Don’t care</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>PC13 Pin function</td>
<td>OSEL[1:0] (ALAR output enable)</td>
<td>TAMPOE (TAMPER output enable)</td>
<td>COE (CALIB output enable)</td>
<td>OUTEN</td>
<td>TAMPALRM_TYPE</td>
<td>TAMPALRM_PU</td>
<td>TAMPE2 &amp; TAMPE2AM</td>
<td>TAMPE3 &amp; TAMPE3AM &amp; OUT3_RMP</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------</td>
<td>-------------------------------</td>
<td>---------------------------</td>
<td>-------</td>
<td>---------------</td>
<td>-------------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>No pull</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>TAMPALRM output</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open-Drain(2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal pull-up</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>CALIB output PP</td>
<td>00</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>TAMP_OUT2 output PP</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>Don’t care</td>
</tr>
<tr>
<td>TAMP_OUT3 output PP</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TAMP_IN1 input floating</td>
<td>00</td>
<td>0</td>
<td>Don’t care</td>
<td></td>
<td></td>
<td></td>
<td>Don’t care</td>
<td>0</td>
</tr>
<tr>
<td>RTC_TS and TAMP_IN1 input floating</td>
<td>00</td>
<td>0</td>
<td>Don’t care</td>
<td></td>
<td></td>
<td></td>
<td>Don’t care</td>
<td>0</td>
</tr>
</tbody>
</table>
| RTC_TS input floating | 00                       | 0                             | Don’t care                |       |               |             | Don’t care        | 0                         | 0                          | 0                         | 1
The output mechanism follows the priority order shown in Table 348.

### Table 347. RTC pin PC13 configuration (continued)

<table>
<thead>
<tr>
<th>PC13 Pin function</th>
<th>OSEL[1:0] (ALARM output enable)</th>
<th>TAMPOE (TAMPER output enable)</th>
<th>COE (CALIB output enable)</th>
<th>OUT2EN</th>
<th>TAMPALRM_TYPE</th>
<th>TAMPALRM_PU</th>
<th>TAMPAE &amp; TAMPAAM (OUT3_RMP = 0)</th>
<th>TAMPE (TAMP_IN1 input enable)</th>
<th>TSE (RTC_TS input enable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wakeup pin or Standard GPIO</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>Don't care</td>
<td>Don't care</td>
<td>Don't care</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. OD: open drain; PP: push-pull.
2. In this configuration the GPIO must be configured in input.

RTC_OUT2, TAMPA2_IN, TAMPA3_OUT are mapped on the same pin (PI8). PI8 configuration is controlled by the RTC, whatever the PI8 GPIO configuration. The RTC or TAMP functions mapped on PI8 are available in all low-power modes and in VBAT mode.

### Table 348. PI8 configuration

<table>
<thead>
<tr>
<th>PI8 Pin function</th>
<th>OSEL[1:0] (ALARM output enable)</th>
<th>TAMPOE (TAMPER output enable)</th>
<th>COE (CALIB output enable)</th>
<th>OUT2EN &amp; OUT2_RMP</th>
<th>TAMPALRM_TYPE</th>
<th>TAMPALRM_PU</th>
<th>TAMPAE &amp; TAMPAAM (OUT3_RMP = 0)</th>
<th>TAMPE (TAMP_IN1 input enable)</th>
<th>TAMP2E</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMPALRM output Push-Pull</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
</tbody>
</table>
In addition, it is possible to output RTC_OUT2 on PB2 or PI8 pin thanks to OUT2EN bit. This output is not available in VBAT mode. The different functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, COE and OUT2EN configuration, as shown in Table 349.

For PB2, the GPIO should be configured as an alternate function. This is not the case for PI8, controlled directly by the RTC.

<table>
<thead>
<tr>
<th>PI8 Pin function</th>
<th>OSEL[1:0] (ALARM output enable)</th>
<th>TAMPOE (TEMPER output enable)</th>
<th>COE (CALIB output enable)</th>
<th>OUT2EN &amp; OUT2_RMP</th>
<th>TAMPALRM_TYPE</th>
<th>TAMPALRM_PU</th>
<th>TAMP3E &amp; TAMP3AM &amp; (OUT3_RMP=0)</th>
<th>TAMP2E</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMPALRM output Open-Drain</td>
<td>No pull</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>Internal pull-up</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>CALIB output PP</td>
<td>00</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>TAMP_OUT3 output Push-Pull</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>TAMP_IN2 input floating</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Wakeup pin</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Standard GPIO</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
</tr>
</tbody>
</table>
50.3.4 Clock and prescalers

The RTC clock source (RTCCLOCK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to Section 8: Reset and clock control (RCC).

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see Figure 503: RTC block diagram):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is $2^{22}$.

This corresponds to a maximum input frequency of around 4 MHz.

$f_{ck\_apre}$ is given by the following formula:

$$f_{CK\_APRE} = \frac{f_{RTCCLOCK}}{PREDIV\_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subseconds downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

$f_{ck\_spre}$ is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLOCK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$
The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see Section 50.3.7: Periodic auto-wakeup for details).

50.3.5 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see Section 50.6.10: RTC shift control register (RTC_SHIFTR)). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

50.3.6 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC_CR register.
50.3.7 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.

The wake-up timer clock input ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.
  When RTCCLK is LSE (32.768 kHz), this allows to configure the wake-up interrupt period from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock)
  When ck_spre frequency is 1 Hz, this allows to achieve a wake-up time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case $2^{16}$ is added to the 16-bit counter current value. When the initialization sequence is complete (see Programming the wake-up timer on page 1828), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wake-up flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop and Standby) have no influence on the wake-up timer.

50.3.8 RTC initialization and configuration

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD = 0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, some of the RTC registers are write-protected.

Writing to the protected RTC registers is enabled by writing a key into the Write Protection register, RTC_WPR.
The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

**Calendar initialization and configuration**

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

**Note:** After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

**Caution:** The active tampers must be disabled before entering the RTC initialization mode (INIT = 1 in the RTC_ICSR). Refer to Section : Active tamper detection for more details.

**Daylight saving time**

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

**Programming the alarm**

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASR/RTC_ALRMAR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.
**Note:** Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

### Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. This step must be skipped in calendar initialization mode. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wakeup auto-reload value WUT[15:0], and the wakeup clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wakeup timer restarts down-counting. The WUTWF bit is cleared up to 2 RTCCLK clocks cycles after WUTE is cleared, due to clock synchronization.

### 50.3.9 Reading the calendar

#### When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB1 clock frequency ($f_{PCLK}$) must be equal to or greater than seven times the RTC clock frequency ($f_{RTCCLK}$). This ensures a secure behavior of the synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycles. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wakeup and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to [Calendar initialization and configuration on page 1827](#)): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to [Section 50.3.11: RTC synchronization](#)): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.
When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

50.3.10 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wakeup timer register (RTC_WUTR), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR), and the configuration register (RTC_CFGR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to RCC for details about RTC clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

50.3.11 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by "shifting" its clock by a fraction of a second using RTC_SHIFTR.

RTC_SSR contains the value of the synchronous prescaler counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]. The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption.
The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of \(1 / (\text{PREDIV}_S + 1)\) seconds. The shift operation consists of adding the \text{SUBFS}[14:0] value to the synchronous prescaler counter \text{SS}[15:0]: this will delay the clock. If at the same time the ADD1S bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this will advance the clock.

**Caution:** Before initiating a shift operation, the user must check that \text{SS}[15] = 0 in order to ensure that no overflow will occur.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

**Caution:** This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

### 50.3.12 RTC reference clock detection

The update of the RTC calendar can be synchronized to a reference clock, RTC\_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC\_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC\_REFIN detection is enabled (REFCKON bit of RTC\_CR set to 1), the calendar is still clocked by the LSE, and RTC\_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC\_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck\_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 \text{ck\_apre} periods when detecting the first reference clock edge. A smaller window of 3 \text{ck\_apre} periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the \text{ck\_spre} clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 \text{ck\_apre} window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 \text{ck\_apre} period detection window centered on the \text{ck\_spre} edge.

When the RTC\_REFIN detection is enabled, \text{PREDIV}_A and \text{PREDIV}_S must be set to their default values:

- \text{PREDIV}_A = 0x007F
- \text{PREDIV}_S = 0x00FF

**Note:** RTC\_REFIN clock detection is not available in Standby mode.
50.3.13 **RTC smooth digital calibration**

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about $2^{20}$ RTCCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, cal_cnt[19:0], clocked by RTCCCLK.

The smooth calibration register (RTC_CALR) specifies the number of RTCCCLK clock cycles to be masked during the 32-second cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

**Note:** CALM[8:0] (RTC_CALR) specifies the number of RTCCCLK pulses to be masked during the 32-second cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra RTCCCLK pulse every $2^{11}$ RTCCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCCLK cycles can be added during the 32-second cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency ($F_{CAL}$) given the input frequency ($F_{RTCCLK}$) is as follows:

$$F_{CAL} = F_{RTCCLK} \times \left[1 + (\text{CALP} \times 512 - \text{CALM}) / (2^{20} + \text{CALM} - \text{CALP} \times 512)\right]$$

**Calibration when PREDIV_A < 3**

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV_A less than 3, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 RTCCCLK clock cycles, which is equivalent to adding 256 clock cycles every 32 seconds. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each 32-second cycle using only the CALM bits.

With a nominal RTCCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other
interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times \left[1 + \frac{256 - CALM}{220 + CALM - 256}\right]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

**Verifying the RTC calibration**

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16-second calibration cycle period.

In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

**Re-calibration on-the-fly**

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

**50.3.14 Timestamp function**

Timestamp is enabled by setting the TSE or ITSE bits of RTC_CR register to 1.
When TSE is set:
The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS is set:
The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a tamper event is detected on the TAMP_INx pin.

When ITSE is set:
The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the VBAT supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

**Note:** TSF is set 2 ck_apre cycles after the timestamp event occurs due to synchronization process.

There is no delay in the setting of TSOVF. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

**Caution:** If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

Optionally, a tamper event can cause a timestamp to be recorded. See the description of the TAMPTS control bit in the RTC control register (RTC_CR).

### 50.3.15 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is \( f_{RTCCLK/64} \). This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is \( f_{RTCCLK/(256 \times (PREDIV_A+1))} \). This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = 0x7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.
Note: When the CALIB output is selected, the RTC_OUT1 pin is automatically configured but the RTC_OUT2 pin must be set as alternate function.
When COSEL is cleared, the CALIB output is the output of the 6th stage of the asynchronous prescaler.
When COSEL is set, the CALIB output is the output of the 8th stage of the synchronous prescaler.

50.3.16 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC SR register.

When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wakeup flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note: Once the TAMPALRM output is enabled, it has priority over CALIB on RTC_OUT1.

When TAMPALRM output is selected, the RTC_OUT1 pin is automatically configured but the RTC_OUT2 pin must be set as alternate function. In case the TAMPALRM is configured open-drain in the RTC, the RTC_OUT1 GPIO must be configured as input.

50.4 RTC low-power modes

Table 350. Effect of low-power modes on RTC

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. RTC interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.</td>
</tr>
</tbody>
</table>

The table below summarizes the RTC pins and functions capability in all modes.
### 50.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register. The interrupt output is also activated.

<table>
<thead>
<tr>
<th>interrupt acronym</th>
<th>interrupt event</th>
<th>event flag</th>
<th>enable control bit</th>
<th>interrupt clear method</th>
<th>exit from sleep mode</th>
<th>exit from stop and standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_Alarm</td>
<td>Alarm A</td>
<td>ALRAF</td>
<td>ALRAIE</td>
<td>write 1 in CALRAF</td>
<td>Yes</td>
<td>Yes(3)</td>
</tr>
<tr>
<td>RTC_Alarm</td>
<td>Alarm B</td>
<td>ALRBF</td>
<td>ALRBIE</td>
<td>write 1 in CALRBF</td>
<td>Yes</td>
<td>Yes(3)</td>
</tr>
<tr>
<td>RTC_TAMP_STAMP_CSS_LSE</td>
<td>Timestamp</td>
<td>TSF</td>
<td>TSIE</td>
<td>write 1 in CTSF</td>
<td>Yes</td>
<td>Yes(3)</td>
</tr>
<tr>
<td>RTC_WKUP</td>
<td>Wakeup timer interrupt</td>
<td>WUTF</td>
<td>WUTIE</td>
<td>write 1 in CWUTF</td>
<td>Yes</td>
<td>Yes(3)</td>
</tr>
</tbody>
</table>

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. Wakeup from Stop and Standby modes is possible only when the RTC clock source is LSE or LSI.

### 50.6 RTC registers

Refer to Section 1.2 on page 97 of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).
50.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to Calendar initialization and configuration on page 1827 and Reading the calendar on page 1828.

This register is write protected. The write access procedure is described in RTC register write protection on page 1826.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

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

Bit 22 PM: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 HT[1:0]: Hour tens in BCD format

Bits 19:16 HU[3:0]: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 MNT[2:0]: Minute tens in BCD format

Bits 11:8 MNU[3:0]: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 ST[2:0]: Second tens in BCD format

Bits 3:0 SU[3:0]: Second units in BCD format
50.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to Calendar initialization and configuration on page 1827 and Reading the calendar on page 1828.

This register is write protected. The write access procedure is described in RTC register write protection on page 1826.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

- Bits 31:24 Reserved, must be kept at reset value.
- Bits 23:20 **YT[3:0]**: Year tens in BCD format
- Bits 19:16 **YU[3:0]**: Year units in BCD format
- Bits 15:13 **WDU[2:0]**: Week day units
  - 000: forbidden
  - 001: Monday
  - …
  - 111: Sunday
- Bit 12 **MT**: Month tens in BCD format
- Bits 11:8 **MU[3:0]**: Month units in BCD format
- Bits 7:6 Reserved, must be kept at reset value.
- Bits 5:4 **DT[1:0]**: Date tens in BCD format
- Bits 3:0 **DU[3:0]**: Date units in BCD format

**Note:** The calendar is frozen when reaching the maximum value, and can't roll over.
50.6.3 RTC sub second register (RTC_SSR)

Address offset: 0x08
Backup domain reset value: 0x0000 0000
System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

**SS[15:0]**

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SS[15:0]:** Sub second value

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

*Note:* SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

50.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in RTC register write protection on page 1826.

Address offset: 0x0C
Backup domain reset value: 0x0000 0007
System reset value: not affected (except INIT, INITF, and RSF bits which are cleared to 0)

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

**RECALPF**

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

Bit 16 **RECALPF:** Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to Re-calibration on-the-fly.

Bits 15:8 Reserved, must be kept at reset value.
Bit 7 INIT: Initialization mode
0: Free running mode
1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 INITF: Initialization flag
When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.
0: Calendar registers update is not allowed
1: Calendar registers update is allowed

Bit 5 RSF: Registers synchronization flag
This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSRx, RTC_TRx and RTC_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPHASD = 1). This bit can also be cleared by software.
0: Calendar shadow registers not yet synchronized
1: Calendar shadow registers synchronized

Bit 4 INITS: Initialization status flag
This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).
0: Calendar has not been initialized
1: Calendar has been initialized

Bit 3 SHPF: Shift operation pending
This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.
0: No shift operation is pending
1: A shift operation is pending

Bit 2 WUTF: Wakeup timer write flag
This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.
It is cleared by hardware in initialization mode.
0: Wakeup timer configuration update not allowed except in initialization mode
1: Wakeup timer configuration update allowed

Bit 1 ALRGBW: Alarm B write flag
This bit is set by hardware when alarm B values can be changed, after the ALRBE bit has been set to 0 in RTC_CR.
It is cleared by hardware in initialization mode.
0: Alarm B update not allowed
1: Alarm B update allowed

Bit 0 ALRAWF: Alarm A write flag
This bit is set by hardware when alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC_CR.
It is cleared by hardware in initialization mode.
0: Alarm A update not allowed
1: Alarm A update allowed
50.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to Calendar initialization and configuration on page 1827.

This register is write protected. The write access procedure is described in RTC register write protection on page 1826.

Address offset: 0x10
Backup domain reset value: 0x007F 00FF
System reset: not affected

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

Bits 31:23 Reserved, must be kept at reset value.
Bits 22:16 PREDIV_A[6:0]: Asynchronous prescaler factor
This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)
Bit 15 Reserved, must be kept at reset value.
Bits 14:0 PREDIV_S[14:0]: Synchronous prescaler factor
This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

50.6.6 RTC wakeup timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.
This register is write protected. The write access procedure is described in RTC register write protection on page 1826.

Address offset: 0x14
Backup domain reset value: 0x0000 FFFF
System reset: not affected

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

WUT[15:0]
**50.6.7 RTC control register (RTC_CR)**

This register is write protected. The write access procedure is described in RTC register write protection on page 1826.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>OUT2EN</th>
<th>Bits 28:27</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 30</td>
<td>TAMPMafety_TYPE</td>
<td>TAMPAoutput type</td>
<td>0: TAMPAoutput is push-pull output 1: TAMPAoutput is open-drain output</td>
</tr>
<tr>
<td>Bit 29</td>
<td>TAMPAoutput_PULL</td>
<td>TAMPAoutput pull-up enable</td>
<td>0: No pull-up is applied on TAMPAoutput 1: A pull-up is applied on TAMPAoutput</td>
</tr>
</tbody>
</table>

**Bits 31:16**
Reserved, must be kept at reset value.

**Bits 15:0**

**WUT[15:0]: Wakeup auto-reload value bits**

- When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) \( \text{ck}_wut \) cycles. The \( \text{ck}_wut \) period is selected through WUCKSEL[2:0] bits of the RTC_CR register.
- When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.
- The first assertion of WUTF occurs between WUT and (WUT + 1) \( \text{ck}_wut \) cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>WUT[15:0]</th>
<th>Wakeup auto-reload value bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>15:0</td>
<td>WUT[15:0]</td>
<td>Wakeup auto-reload value bits</td>
</tr>
</tbody>
</table>

**Bit 31**

**OUT2EN**: RTC_OUT2 output enable

- Setting this bit allows to remap the RTC outputs on RTC_OUT2 as follows:
  - OUT2EN = 0: RTC output 2 disable
  - If OSEL # 00 or TAMPOE = 1: TAMPAoutput is output on RTC_OUT1
  - If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1
  - OUT2EN = 1: RTC output 2 enable
  - If (OSEL # 00 or TAMPOE = 1) and COE = 0: TAMPAoutput is output on RTC_OUT2
  - If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2
  - If (OSEL # 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPAoutput is output on RTC_OUT1.

**Bit 30**

**TAMPAoutput_TYPE**: TAMPAoutput type

- 0: TAMPAoutput is push-pull output
- 1: TAMPAoutput is open-drain output

**Bit 29**

**TAMPAoutput_PULL**: TAMPAoutput pull-up enable

- 0: No pull-up is applied on TAMPAoutput
- 1: A pull-up is applied on TAMPAoutput

**Bits 28:27**
Reserved, must be kept at reset value.
Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM  
0: The tamper flag is not routed on TAMPALRM  
1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event  
0: Tamper detection event does not cause a RTC timestamp to be saved  
1: Save RTC timestamp on tamper detection event  
TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set after the tamper flags, therefore if TAMPTS and TSIE are set, it is recommended to disable the tamper interrupts in order to avoid servicing 2 interrupts.

Bit 24 **ITSE**: timestamp on internal event enable  
0: internal event timestamp disabled  
1: internal event timestamp enabled

Bit 23 **COE**: Calibration output enable  
This bit enables the CALIB output  
0: Calibration output disabled  
1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection  
These bits are used to select the flag to be routed to TAMPALRM output.  
00: Output disabled  
01: Alarm A output enabled  
10: Alarm B output enabled  
11: Wakeup output enabled

Bit 20 **POL**: Output polarity  
This bit is used to configure the polarity of TAMPALRM output.  
0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).  
1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection  
When COE = 1, this bit selects which signal is output on CALIB.  
0: Calibration output is 512 Hz  
1: Calibration output is 1 Hz  
These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to Section 50.3.15: Calibration clock output.

Bit 18 **BKP**: Backup  
This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)  
When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.  
Setting this bit has no effect when current hour is 0.  
0: No effect  
1: Subtracts 1 hour to the current time. This can be used for winter time change.
Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.

0: No effect
1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable

0: Timestamp interrupt disable
1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wakeup timer interrupt enable

0: Wakeup timer interrupt disabled
1: Wakeup timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable

0: Alarm B interrupt disable
1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable

0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable

0: timestamp disable
1: timestamp enable

Bit 10 **WUTE**: Wakeup timer enable

0: Wakeup timer disabled
1: Wakeup timer enabled

*Note: When the wakeup timer is disabled, wait for WUTWF=1 before enabling it again.*

Bit 9 **ALRBE**: Alarm B enable

0: Alarm B disabled
1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable

0: Alarm A disabled
1: Alarm A enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FMT**: Hour format

0: 24 hour/day format
1: AM/PM hour format

Bit 5 **BYPSSHAD**: Bypass the shadow registers

0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.
1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.

*Note: If the frequency of the APB1 clock is less than seven times the frequency of RTCCLK, BYPSSHAD must be set to 1.*
Bit 4 **REFCKON**: RTC\_REFIN reference clock detection enable (50 or 60 Hz)
- 0: RTC\_REFIN detection disabled
- 1: RTC\_REFIN detection enabled

*Note: PREDIV\_S must be 0x00FF.*

Bit 3 **TSEDGE**: Timestamp event active edge
- 0: RTC\_TS input rising edge generates a timestamp event
- 1: RTC\_TS input falling edge generates a timestamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck\_wut wakeup clock selection
- 000: RTC/16 clock is selected
- 001: RTC/8 clock is selected
- 010: RTC/4 clock is selected
- 011: RTC/2 clock is selected
- 10x: ck\_spre (usually 1 Hz) clock is selected
- 11x: ck\_spre (usually 1 Hz) clock is selected and $2^{16}$ is added to the WUT counter value

*Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC\_ICSR/INITF = 1).*

WUT = wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC\_CR WUTE bit = 0 and RTC\_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

### 50.6.8 RTC write protection register (RTC\_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to **RTC register write protection** for a description of how to unlock RTC register write protection.
50.6.9  RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in RTC register write protection on page 1826.

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CALP</td>
<td>CALW8</td>
<td>CALW16</td>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td>CALM[8:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bit 15  CALP: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.
1: One RTCCLK pulse is effectively inserted every $2^{11}$ pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. If the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows: $(512 \times \text{CALP}) - \text{CALM}$.

Refer to Section 50.3.13: RTC smooth digital calibration.

Bit 14  CALW8: Use an 8-second calibration cycle period

When CALW8 is set to 1, the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to Section 50.3.13: RTC smooth digital calibration.

Bit 13  CALW16: Use a 16-second calibration cycle period

When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.

Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 50.3.13: RTC smooth digital calibration.

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

Bits 8:0  CALM[8:0]: Calibration minus

The frequency of the calendar is reduced by masking CALM out of $2^{20}$ RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See Section 50.3.13: RTC smooth digital calibration on page 1831.
50.6.10 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in RTC register write protection on page 1826.

Address offset: 0x2C
Backup domain reset value: 0x0000 0000
System reset: not affected

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

Bit 31 **ADD1S**: Add one second
0: No effect
1: Add one second to the clock/calendar
This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).
This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second
These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).
The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:
Delay (seconds) = SUBFS / (PREDIV_S + 1)
A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:
Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.
50.6.11  **RTC timestamp time register (RTC_TSTR)**

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

- **Address offset:** 0x30
- **Backup domain reset value:** 0x0000 0000
- **System reset:** not affected

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

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

- **Bit 22**  PM: AM/PM notation
  - 0: AM or 24-hour format
  - 1: PM

- **Bits 21:20**  HT[1:0]: Hour tens in BCD format.
- **Bits 19:16**  HU[3:0]: Hour units in BCD format.
  - **Bit 15**  Reserved, must be kept at reset value.

- **Bits 14:12**  MNT[2:0]: Minute tens in BCD format.

- **Bits 11:8**  MNU[3:0]: Minute units in BCD format.
  - **Bit 7**  Reserved, must be kept at reset value.

- **Bits 6:4**  ST[2:0]: Second tens in BCD format.
- **Bits 3:0**  SU[3:0]: Second units in BCD format.

50.6.12  **RTC timestamp date register (RTC_TSDR)**

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

- **Address offset:** 0x34
- **Backup domain reset value:** 0x0000 0000
- **System reset:** not affected

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

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

- **Bit 22**  WDU[2:0]: Weekday tens in BCD format.
- **Bit 21**  MT: Month
  - 1: January
  - 2: February
  - 3: March
  - 4: April
  - 5: May
  - 6: June
  - 7: July
  - 8: August
  - 9: September
  - 10: October
  - 11: November
  - 12: December
- **Bit 20**  MJU[3:0]: Month units in BCD format.
  - **Bit 19**  Reserved, must be kept at reset value.

- **Bits 18:16**  ST[2:0]: Second tens in BCD format.
- **Bits 15:12**  DT[1:0]: Date in BCD format.
- **Bits 11:8**  DU[3:0]: Day units in BCD format.
Bits 31:16  Reserved, must be kept at reset value.

Bits 15:13  **WDU[2:0]**: Week day units

Bit 12  **MT**: Month tens in BCD format

Bits 11:8  **MU[3:0]**: Month units in BCD format

Bits 7:6  Reserved, must be kept at reset value.

Bits 5:4  **DT[1:0]**: Date tens in BCD format

Bits 3:0  **DU[3:0]**: Date units in BCD format

---

**50.6.13  RTC timestamp sub second register (RTC_TSSSR)**

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

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

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

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **SS[15:0]**: Sub second value

**SS[15:0]** is the value of the synchronous prescaler counter when the timestamp event occurred.
50.6.14 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in RTC register write protection on page 1826.

Address offset: 0x40
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>MSK4: Alarm A date mask</td>
<td>0: Alarm A set if the date/day match</td>
</tr>
<tr>
<td>30</td>
<td>WDSEL: Week day selection</td>
<td>0: DU[3:0] represents the date units</td>
</tr>
<tr>
<td>29</td>
<td>DT[1:0]: Date tens in BCD format</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>DU[3:0]: Date units or day in BCD format</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>MSK3: Alarm A hours mask</td>
<td>0: Alarm A set if the hours match</td>
</tr>
<tr>
<td>26</td>
<td>PM: AM/PM notation</td>
<td>0: AM or 24-hour format</td>
</tr>
<tr>
<td>25</td>
<td>HT[1:0]: Hour tens in BCD format</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>HU[3:0]: Hour units in BCD format</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>MSK2: Alarm A minutes mask</td>
<td>0: Alarm A set if the minutes match</td>
</tr>
<tr>
<td>22</td>
<td>ST[2:0]: Second tens in BCD format</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>SU[3:0]: Second units in BCD format</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 MSK4: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don’t care in alarm A comparison

Bit 30 WDSEL: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28 DT[1:0]: Date tens in BCD format
Bits 27:24 DU[3:0]: Date units or day in BCD format

Bit 23 MSK3: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don’t care in alarm A comparison

Bit 22 PM: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 HT[1:0]: Hour tens in BCD format
Bits 19:16 HU[3:0]: Hour units in BCD format

Bit 15 MSK2: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don’t care in alarm A comparison

Bits 14:12 MNT[2:0]: Minute tens in BCD format
Bits 11:8 MNU[3:0]: Minute units in BCD format

Bit 7 MSK1: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don’t care in alarm A comparison

Bits 6:4 ST[2:0]: Second tens in BCD format.
Bits 3:0 SU[3:0]: Second units in BCD format.
### 50.6.15 RTC alarm A sub second register (RTC_ALRMASSR)

This register can be written only when ALRAWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1826*.

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>MASKSS[3:0]: Mask the most-significant bits starting at this bit</td>
<td>rw</td>
<td>0x0</td>
</tr>
<tr>
<td>22</td>
<td>0: No comparison on sub seconds for alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1: SS[14:1] are don’t care in alarm A comparison. Only SS[0] is compared.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>2: SS[14:2] are don’t care in alarm A comparison. Only SS[1:0] are compared.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>15: All 15 SS bits are compared and must match to activate alarm. The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Note: The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>SS[14:0]: Sub seconds value</td>
<td>rw</td>
<td>0x0</td>
</tr>
<tr>
<td>10</td>
<td>This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 27:24 MASKSS[3:0]: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[14:1] are don’t care in alarm A comparison. Only SS[0] is compared.

2: SS[14:2] are don’t care in alarm A comparison. Only SS[1:0] are compared.


... 12: SS[14:12] are don’t care in alarm A comparison. SS[11:0] are compared.


15: All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Note: The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 SS[14:0]: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.
50.6.16  RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in RTC register write protection on page 1826.

Address offset: 0x48
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>MSK4: Alarm B date mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Alarm B set if the date and day match</td>
<td></td>
</tr>
<tr>
<td>1: Date and day don’t care in alarm B comparison</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>WDSEL: Week day selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: DU[3:0] represents the date units</td>
<td></td>
</tr>
<tr>
<td>1: DU[3:0] represents the week day. DT[1:0] is don’t care.</td>
<td></td>
</tr>
</tbody>
</table>

| Bits 29:28 | DT[1:0]: Date tens in BCD format |
| Bits 27:24 | DU[3:0]: Date units or day in BCD format |

<table>
<thead>
<tr>
<th>Bit 23</th>
<th>MSK3: Alarm B hours mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Alarm B set if the hours match</td>
<td></td>
</tr>
<tr>
<td>1: Hours don’t care in alarm B comparison</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 22</th>
<th>PM: AM/PM notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: AM or 24-hour format</td>
<td></td>
</tr>
<tr>
<td>1: PM</td>
<td></td>
</tr>
</tbody>
</table>

| Bits 21:20 | HT[1:0]: Hour tens in BCD format |
| Bits 19:16 | HU[3:0]: Hour units in BCD format |

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>MSK2: Alarm B minutes mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Alarm B set if the minutes match</td>
<td></td>
</tr>
<tr>
<td>1: Minutes don’t care in alarm B comparison</td>
<td></td>
</tr>
</tbody>
</table>

| Bits 14:12 | MNT[2:0]: Minute tens in BCD format |
| Bits 11:8  | MNU[3:0]: Minute units in BCD format |

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>MSK1: Alarm B seconds mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Alarm B set if the seconds match</td>
<td></td>
</tr>
<tr>
<td>1: Seconds don’t care in alarm B comparison</td>
<td></td>
</tr>
</tbody>
</table>

| Bits 6:4  | ST[2:0]: Second tens in BCD format |
| Bits 3:0  | SU[3:0]: Second units in BCD format |
### 50.6.17 RTC alarm B sub second register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in Section : RTC register write protection.

Address offset: 0x4C
Backup domain reset value: 0x0000 0000
System reset: not affected

| Bits 31:28 Reserved, must be kept at reset value. |
|---|---|
| Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit |
| 0x0: No comparison on sub seconds for alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match). |
| 0x1: SS[14:1] are don’t care in alarm B comparison. Only SS[0] is compared. |
| 0x2: SS[14:2] are don’t care in alarm B comparison. Only SS[1:0] are compared. |
| 0x3: SS[14:3] are don’t care in alarm B comparison. Only SS[2:0] are compared. |
| ... |
| 0xC: SS[14:12] are don’t care in alarm B comparison. SS[11:0] are compared. |
| 0xD: SS[14:13] are don’t care in alarm B comparison. SS[12:0] are compared. |
| 0xE: SS[14] is don’t care in alarm B comparison. SS[13:0] are compared. |
| 0xF: All 15 SS bits are compared and must match to activate alarm. |
| The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation. |

| Bits 23:15 Reserved, must be kept at reset value. |
|---|---|
| Bits 14:0 **SS[14:0]**: Sub seconds value |
| This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared. |

### 50.6.18 RTC status register (RTC_SR)

Address offset: 0x50
Backup domain reset value: 0x0000 0000
System reset: not affected
Real-time clock (RTC)

Note: The bits of this register are cleared 2 APB clock cycles after setting their corresponding clear bit in the RTC_SCR register.

50.6.19 RTC masked interrupt status register (RTC_MISR)

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected
50.6.20 RTC status clear register (RTC_SCR)

Address offset: 0x5C
Backup domain reset value: 0x0000 0000
System reset: not affected

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

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **CITSF**: Clear internal timestamp flag
Writing 1 in this bit clears the ITSF bit in the RTC_SR register.

Bit 4 **CTSOVF**: Clear timestamp overflow flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.
Bit 2  **CWUTF**: Clear wakeup timer flag  
Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1  **CALRB**: Clear alarm B flag  
Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0  **CALRA**: Clear alarm A flag  
Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

### 50.6.21 RTC configuration register (RTC_CFRG)

Address offset: 0x60  
Backup domain reset value: 0x0000 0000  
System reset: not affected

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

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

Bit 0  **OUT2_RMP**: RTC_OUT2 mapping  
0: RTC_OUT2 is mapped on PB2  
1: RTC_OUT2 is mapped on PI8
Table 353. RTC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset Register</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>RTC_TR</td>
<td>Offset Register</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x04</td>
<td>RTC_DR</td>
<td>Offset Register</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x08</td>
<td>RTC_SSR</td>
<td>Offset Register</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0C</td>
<td>RTC_ICSR</td>
<td>Offset Register</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x10</td>
<td>RTC_PRER</td>
<td>Offset Register</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x14</td>
<td>RTC_WUTR</td>
<td>Offset Register</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x18</td>
<td>RTC_CR</td>
<td>Offset Register</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x24</td>
<td>RTC_WPR</td>
<td>Offset Register</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x28</td>
<td>RTC_CALR</td>
<td>Offset Register</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x2C</td>
<td>RTC_SHIFTTR</td>
<td>Offset Register</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x30</td>
<td>RTC_TSTR</td>
<td>Offset Register</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x34</td>
<td>RTC_TSDR</td>
<td>Offset Register</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x38</td>
<td>RTC_TSSSR</td>
<td>Offset Register</td>
<td>Reset value</td>
</tr>
</tbody>
</table>
Refer to Section 2.3 on page 128 for the register boundary addresses.
51 Tamper and backup registers (TAMP)

51.1 Introduction

32 32-bit backup registers are retained in all low-power modes and also in \( V_{\text{BAT}} \) mode. They can be used to store sensitive data as their content is protected by an tamper detection circuit. 3 tamper pins and 7 internal tampers are available for anti-tamper detection. The external tamper pins can be configured for edge detection, or level detection with or without filtering, or active tamper which increases the security level by auto checking that the tamper pins are not externally opened or shorted.

51.2 TAMP main features

- 32 backup registers:
  - the backup registers (TAMP_BKPxR) are implemented in the RTC domain that remains powered-on by \( V_{\text{BAT}} \) when the \( V_{\text{DD}} \) power is switched off.
- 3 external tamper detection events.
  - Each external event can be configured to be active or passive.
  - External passive tampers with configurable filter and internal pull-up.
- 7 internal tamper events.
- Any tamper detection can generate a RTC timestamp event.
- Any tamper detection can erase the backup registers, backup SRAM and OTFDEC keys & CRC registers.
- Monotonic counter.
51.3 TAMP functional description

51.3.1 TAMP block diagram

Figure 504. TAMP block diagram

1. The number of external and internal tampers depends on products.
51.3.2 TAMP pins and internal signals

Table 354. TAMP input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP_INx (x = pin index)</td>
<td>Input</td>
<td>Tamper input pin</td>
</tr>
<tr>
<td>TAMP_OUTx (x = pin index)</td>
<td>Output</td>
<td>Tamper output pin</td>
</tr>
</tbody>
</table>

Table 355. TAMP internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tamp_ker_ck</td>
<td>Input</td>
<td>TAMP kernel clock, connected to rtc_ker_ck and also named RTCCCLK in this document</td>
</tr>
<tr>
<td>tamp_pclk</td>
<td>Input</td>
<td>TAMP APB clock, connected to rtc_pclk</td>
</tr>
<tr>
<td>tamp_itamp[y] (y = signal index)</td>
<td>Inputs</td>
<td>Internal tamper event sources</td>
</tr>
<tr>
<td>tamp_evt</td>
<td>Output</td>
<td>Tamper event detection (internal or external) The tamp_evt is used to generate a RTC timestamp event</td>
</tr>
<tr>
<td>tamp_erase</td>
<td>Output</td>
<td>Device secrets erase request following tamper event detection (internal or external)</td>
</tr>
<tr>
<td>tamp_it</td>
<td>Output</td>
<td>TAMP interrupt (refer to Section 51.5: TAMP interrupts for details)</td>
</tr>
<tr>
<td>tamp_trg[x] (x = signal index)</td>
<td>Output</td>
<td>Tamper detection trigger</td>
</tr>
</tbody>
</table>

Table 356. TAMP interconnection

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>tamp_evt</td>
<td>rtc_tamp_evt used to generate a timestamp event</td>
</tr>
<tr>
<td>tamp_erase</td>
<td>The tamp_erase signal is used to erase the device secrets listed hereafter: backup registers, backup SRAM and OTFDEC keys &amp; CRC registers</td>
</tr>
<tr>
<td>tamp_itamp1</td>
<td>RTC voltage domain monitoring</td>
</tr>
<tr>
<td>tamp_itamp2</td>
<td>Temperature monitoring</td>
</tr>
<tr>
<td>tamp_itamp3</td>
<td>LSE monitoring</td>
</tr>
<tr>
<td>tamp_itamp4</td>
<td>HSE monitoring</td>
</tr>
<tr>
<td>tamp_itamp5</td>
<td>RTC calendar overflow (rtc_calovf)</td>
</tr>
<tr>
<td>tamp_itamp6</td>
<td>ST manufacturer readout</td>
</tr>
<tr>
<td>tamp_itamp8(1)</td>
<td>Monotonic counter overflow</td>
</tr>
</tbody>
</table>

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some detections modes are not available in some low-power modes or V_{BAT} when the selected clock is not LSE (refer to Section 51.4: TAMP low-power modes for more details).
1. This signal is generated in the TAMP peripheral.

For GPIOs controlled by the TAMP refer to Table 347: RTC pin PC13 configuration and Table 348: PI8 configuration.

51.3.3 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable TAMP registers write access.

51.3.4 Tamper detection

The tamper detection can be configured for the following purposes:

- erase the backup registers and the SRAMs listed in Table 356: TAMP interconnection (default configuration)
- generate an interrupt, capable to wakeup from Stop and Standby mode
- generate a hardware trigger for the low-power timers

TAMP backup registers

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers are reset when a tamper detection event occurs except if the TAMPxNOER bit is set, or if the TAMPxMSK is set in the TAMP_CR2 register.

Note: The backup registers are also erased when the readout protection of the flash is changed from level 1 to level 0.

Tamper detection initialization

Each input can be enabled by setting the corresponding TAMPxEx bits to 1 in the TAMP_CR register.

Each TAMPxINx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

When TAMPxMSK is cleared:

The TAMPxF flag is asserted after the tamper event on the pin, with the latency provided below:

- 3 ck_apre cycles when TAMPFLT differs from 0x0 (level detection with filtering)
- 3 ck_apre cycles when TAMPTS = 1 (timestamp on tamper event)
- No latency when TAMPFLT = 0x0 (edge detection) and TAMPTS = 0

A new tamper occurring on the same pin during this period and as long as TAMPxF is set cannot be detected.

When TAMPxMSK is set:

A new tamper occurring on the same pin cannot be detected during the latency described above and 2.5 ck_rtc additional cycles.
By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

**Trigger output generation on tamper event**

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit in cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration allows to trig automatically the low-power timers in Stop mode, without requiring the system wakeup to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in the **Level detection with filtering on tamper inputs (passive mode)** mode (TAMPFLT ≠ 00 and active mode is not selected).

**Timestamp on tamper event**

With TAMPTS set to 1 in the RTC_CR, any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register TAMPxF is set in the TAMP_SR at the same time that TSF or TSOVF is set in the RTC_SR.

**Edge detection on tamper inputs (passive mode)**

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge/high level or a falling edge/low level is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

**Caution:**

When using the edge detection, it is recommended to check by software the tamper pin level just after enabling the tamper detection (by reading the GPIO registers), and before writing sensitive values in the backup registers, to ensure that an active edge did not occur before enabling the tamper event detection.

When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx should be disabled and then re-enabled (TAMPxIE set to 1) before re-programming the backup registers (TAMP_BKPxR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

**Note:**

Tamper detection is still active when V_DD power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPx is mapped should be externally tied to the correct level.

**Level detection with filtering on tamper inputs (passive mode)**

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.
The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPPFREQ to determine the frequency of the sampling for level detection.

Note: Refer to the datasheet for the electrical characteristics of the pull-up resistors.

Active tamper detection

When the TAMPxAM bit is set in the TAMP_ATCR, the tamper events are configured in active mode, which is based on a comparison between a TAMP_OUTy pin and a TAMP_INx pin. By default (ATOSHARE = 0) the comparison is made between TAMP_INx and TAMP_OUTx (y = x). When ATOSHARE bit is set, the same output can be used for several tamper inputs. Refer to ATOSHARE and ATOSEL bits descriptions in the TAMP_ATCR register.

Every two CK_ATPER cycles (CK_ATPER = 2ATPER x CK_ATPRE = 2ATPER x 2ATCKSEL RTCCLK), TAMP_OUTy output pin provides a value provided by a pseudo random number generator (PRNG). After outputting this value, the TAMP_OUTy pin outputs its opposite value one CK_ATPER cycle after.

PRNG is consumed by the selected tamper outputs at a different frequency depending on the number of selected tamper outputs. The number of selected outputs depends on TAMPxAM, TAMPxE, ATOSEL and ATOSHARE.

- When only 1 output is selected: PRNG is consumed every 16 CK_ATPER periods.
- When 2 outputs are selected: PRNG is consumed every 8 CK_ATPER periods.
- When 3 or 4 outputs are selected: PRNG is consumed every 4 CK_ATPER periods.
- When 5 or more outputs are selected: PRNG is consumed every 2 CK_ATPER periods

The PRNG needs minimum 9 CK_ATPRE cycles to output a new value. Consequently the minimum ATPER values for correct functionality are provided in the table below:

<table>
<thead>
<tr>
<th>Number of selected outputs</th>
<th>Minimum ATPER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3 or 4</td>
<td>2</td>
</tr>
<tr>
<td>5 or more</td>
<td>3</td>
</tr>
</tbody>
</table>

The TAMP_INx pin is externally connected to TAMP_OUTy pin. The comparison is made between TAMP_OUTy output value and TAMP_INx received value, taking into account feedback delay. In case a comparison mismatch occurs, the TAMPxF bit is set in the TAMP_SR register.

As an example, TAMP_OUT1 can be used for comparison with TAMP_IN1 and TAMP_IN2 by configuring and enabling both TAMP1 and TAMP2 in active mode, with ATOSHARE = 1, ATOSEL1 = 00 and ATOSEL2 = 00.
The active tamper can be combined with input filtering when FLTEN = 1. In this case, the tamper is detected only when 2 comparisons are false, in 4 consecutive comparison samples.

The pseudo-random generator must be initially and periodically fed with a new seed. This is done by writing consecutively four 32-bit random values in the TAMP_ATSEEDR register. Programming the seed automatically sends it to the PRNG. As long as the new seed is transferred and elaborated by the PRNG, the SEEDF bit is set in the TAMP_ATOR and it is not allowed to switch off the TAMP APB clock. The duration of the elaboration is up to 184 APB clock cycles after the forth seed is written. Consequently, after writing a new seed, the user must wait until SEEDF is cleared before entering low-power modes.

The active tamper outputs are activated only after the first seed is written and the elaboration is completed. Then new seeds can be written and elaborated during active tamper activity.

**Active tamper initialization**

Here is the software procedure to initialize the active tampers after system reset:

1. Read INITS in TAMP_ATOR register.
2. If INITS = 0x0 (initialization was not done):
   a) Write TAMP_ATCR to configure Active tamper clock, filter and output sharing if any, and active mode.
   b) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
   c) Write SEED by writing four times in the TAMP_ATSEEDR.
   d) Wait until SEEDF = 0 in RTC_ATOR. Backup registers are then protected by active tamper.
3. If INITS = 0x1 (initialization already done):
   No initialization. To increase randomness a new SEED should be provided regularly. When a new SEED is provided, wait until SEEDF = 0 before entering a low-power mode which switches off the RTC APB clock.
4. In case the tampers are disabled by software, and re-enabled afterwards, the SEED must be written after enabling tampers:
   a) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
   b) Write SEED by writing four times in the TAMP_ATSEEDR.
   c) Wait until SEEDF = 0 in RTC_ATOR. Backup registers are then protected by active tamper.

**Caution:** The active tampers must be disabled before entering the RTC initialization mode (INIT = 1 in the RTC_ICSR). Refer to Section : Calendar initialization and configuration for more details.
51.4 TAMP low-power modes

Table 358. Effect of low-power modes on TAMP

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

Table 359. TAMP pins functionality over modes

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Functional in all low-power modes</th>
<th>Functional in V_{BAT} mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP_INx, x = 1,2,3</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>TAMP_OUTx, x = 2,3</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

51.5 TAMP interrupts

The interrupt channel is set in the interrupt status register. The interrupt output is also activated.

Table 360. Interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag(1)</th>
<th>Enable control bit(2)</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop and Standby modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_TAMP_STAMP_CSS_LSE</td>
<td>Tamper x(3)</td>
<td>TAMPxF</td>
<td>TAMPxIE</td>
<td>Write 1 in CTAMPxF</td>
<td>Yes</td>
<td>Yes(4)</td>
</tr>
<tr>
<td></td>
<td>Internal tamper y(3)</td>
<td>ITAMPyF</td>
<td>ITAMPyIE</td>
<td>Write 1 in CITAMPxF</td>
<td>Yes</td>
<td>Yes(4)</td>
</tr>
</tbody>
</table>

1. The event flags are in the TAMP\_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the TAMP\_MISR register.
3. The number of tampers and internal tampers events depend on products.
4. In case of level detection with filtering passive tamper mode, or in case of active tamper mode, wakeup from Stop and Standby modes is possible only when the TAMP clock source is LSE or LSI.

51.6 TAMP registers

Refer to Section 1.2 on page 97 of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).
51.6.1  **TAMP control register 1 (TAMP_CR1)**

Address offset: 0x00

Backup domain reset value: 0xFFFF 0000

System reset: not affected

| Bit 31:24 Reserved, must be kept at reset value. |
| Bit 23 **ITAMP8E**: Internal tamper 8 enable: monotonic counter overflow  
  0: Internal tamper 8 disabled.  
  1: Internal tamper 8 enabled: a tamper is generated when the TAMP_CNT overflows. |
| Bit 22 Reserved, must be kept at reset value. |
| Bit 21 **ITAMP6E**: Internal tamper 6 enable: ST manufacturer readout  
  0: Internal tamper 6 disabled.  
  1: Internal tamper 6 enabled: a tamper is generated in case of ST manufacturer readout. |
| Bit 20 **ITAMP5E**: Internal tamper 5 enable: RTC calendar overflow  
  0: Internal tamper 5 disabled.  
  1: Internal tamper 5 enabled: a tamper is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow. |
| Bit 19 **ITAMP4E**: Internal tamper 4 enable: HSE monitoring  
  0: Internal tamper 4 disabled.  
  1: Internal tamper 4 enabled: a tamper is generated when the HSE frequency is below or above thresholds. |
| Bit 18 **ITAMP3E**: Internal tamper 3 enable: LSE monitoring  
  0: Internal tamper 3 disabled.  
  1: Internal tamper 3 enabled: a tamper is generated when the LSE frequency is below or above thresholds. |
| Bit 17 **ITAMP2E**: Internal tamper 2 enable: Temperature monitoring  
  0: Internal tamper 2 disabled.  
  1: Internal tamper 2 enabled: a tamper is generated when the temperature is below or above thresholds. |
| Bit 16 **ITAMP1E**: Internal tamper 1 enable: RTC power domain supply monitoring  
  0: Internal tamper 1 disabled.  
  1: Internal tamper 1 enabled: a tamper is generated when the RTC power domain supply is below or above thresholds. |

Bits 15:3 Reserved, must be kept at reset value.

---

**Notes:**

- **RM0455**
- **Tamper and backup registers (TAMP)**
- **1866/2974 RM0455 Rev 3**
- **ST**
Bit 2 **TAMP3E**: Tamper detection on TAMP_IN3 enable
0: Tamper detection on TAMP_IN3 is disabled.
1: Tamper detection on TAMP_IN3 is enabled.

Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable
0: Tamper detection on TAMP_IN2 is disabled.
1: Tamper detection on TAMP_IN2 is enabled.

Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable
0: Tamper detection on TAMP_IN1 is disabled.
1: Tamper detection on TAMP_IN1 is enabled.

### 51.6.2 **TAMP control register 2 (TAMP_CR2)**

Address offset: 0x04
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td><strong>TAMP3TRG</strong>: Active level for tamper 3 input (active mode disabled)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 26 **TAMP3TRG**: Active level for tamper 3 input (active mode disabled)
0: If TAMPFLT ≠ 00 Tamper 3 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 3 input rising edge and high level triggers a tamper detection event.
1: If TAMPFLT ≠ 00 Tamper 3 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 3 input falling edge and low level triggers a tamper detection event.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input (active mode disabled)
0: If TAMPFLT ≠ 00 Tamper 2 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 2 input rising edge and high level triggers a tamper detection event.
1: If TAMPFLT ≠ 00 Tamper 2 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 2 input falling edge and low level triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input (active mode disabled)
0: If TAMPFLT ≠ 00 Tamper 1 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 1 input rising edge and high level triggers a tamper detection event.
1: If TAMPFLT ≠ 00 Tamper 1 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 1 input falling edge and low level triggers a tamper detection event.

Bit 23 Reserved, must be kept at reset value.
51.6.3 **TAMP filter control register (TAMP_FLTCR)**

Address offset: 0x0C

Backup domain reset value: 0x0000 0000

System reset: not affected

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

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

- **TAMP3MSK**: Tamper 3 mask
  - 0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.
  - 1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers\(^1\) are not erased.

  *The tamper 3 interrupt must not be enabled when TAMP3MSK is set.*

- **TAMP2MSK**: Tamper 2 mask
  - 0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.
  - 1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers\(^1\) are not erased.

  *The tamper 2 interrupt must not be enabled when TAMP2MSK is set.*

- **TAMP1MSK**: Tamper 1 mask
  - 0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.
  - 1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers\(^1\) are not erased.

  *The tamper 1 interrupt must not be enabled when TAMP1MSK is set.*

Bits 15:3 Reserved, must be kept at reset value.

- **TAMP3NOER**: Tamper 3 no erase
  - 0: Tamper 3 event erases the backup registers.
  - 1: Tamper 3 event does not erase the backup registers\(^1\).

- **TAMP2NOER**: Tamper 2 no erase
  - 0: Tamper 2 event erases the backup registers.
  - 1: Tamper 2 event does not erase the backup registers\(^1\).

- **TAMP1NOER**: Tamper 1 no erase
  - 0: Tamper 1 event erases the backup registers.
  - 1: Tamper 1 event does not erase the backup registers\(^1\).

1. and the device secrets erased by tamp_erase signal (refer to **Table 356: TAMP interconnection**).
1880

Note: This register concerns only the tamper inputs in passive mode.

51.6.4 TAMP active tamper control register 1 (TAMP_ATCR1)

Address offset: 0x10
Backup domain reset value: 0x0007 0000
System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bit 31  **FLTEN**: Active tamper filter enable
- 0: Active tamper filtering disable
- 1: Active tamper filtering enable: a tamper event is detected when 2 comparison mismatches occur out of 4 consecutive samples.

Bit 30  **ATOSHARE**: Active tamper output sharing
- 0: Each active tamper input TAMP_INi is compared with its dedicated output TAMP_OUTi
- 1: Each active tamper input TAMP_INi is compared with TAMPOUTSELx as defined below, with TAMPOUTSELx defined by ATOSELx bits.

Bits 29-27 Reserved, must be kept at reset value.

Bits 26-24  **ATPER[2:0]**: Active tamper output change period
- The tamper output is changed every CK_ATPER = (2^ATPER x CK_ATPRE) cycles. Refer to Table 357: Minimum ATPER value.

Bits 23-19 Reserved, must be kept at reset value.

Bits 18-16  **ATCKSEL[2:0]**: Active tamper RTC asynchronous prescaler clock selection
- These bits select the RTC asynchronous prescaler stage output. The selected clock is CK_ATPRE.
  - f_{CK_ATPRE} = f_{RTCCLK} / 2^{ATCKSEL} when (PREDIV_A+1) = 128.
  - 000: RTCCLK is selected
  - 001: RTCCLK/2 is selected when (PREDIV_A+1) = 128 (actually selects 1\textsuperscript{st} flip flop output)
  - 010: RTCCLK/4 is selected when (PREDIV_A+1) = 128 (actually selects 2\textsuperscript{nd} flip flop output)
  - ...
  - 111: RTCCLK/128 is selected when (PREDIV_A+1) = 128 (actually selects 7\textsuperscript{th} flip flop output)

Note: These bits can be written only when all active tampers are disabled. The write protection remains for up to 1.5 ck_atpre cycles after all the active tampers are disable.

Bits 15-14 Reserved, must be kept at reset value.

Bits 13-12  **ATOSEL3[1:0]**: Active tamper shared output 3 selection
- 00: TAMPOUTSEL3 = TAMP_OUT1
- 01: TAMPOUTSEL3 = TAMP_OUT2
- 10: TAMPOUTSEL3 = TAMP_OUT3
- 11: The selected output must be available in the package pinout

Bits 11-10  **ATOSEL2[1:0]**: Active tamper shared output 2 selection
- 00: TAMPOUTSEL2 = TAMP_OUT1
- 01: TAMPOUTSEL2 = TAMP_OUT2
- 10: TAMPOUTSEL2 = TAMP_OUT3
- 11: The selected output must be available in the package pinout

Bits 9-8  **ATOSEL1[1:0]**: Active tamper shared output 1 selection
- 00: TAMPOUTSEL1 = TAMP_OUT1
- 01: TAMPOUTSEL1 = TAMP_OUT2
- 10: TAMPOUTSEL1 = TAMP_OUT3
- 11: The selected output must be available in the package pinout

Bits 7-3 Reserved, must be kept at reset value.
Bit 2 **TAMP3AM**: Tamper 3 active mode
- 0: Tamper 3 detection mode is passive.
- 1: Tamper 3 detection mode is active.

Bit 1 **TAMP2AM**: Tamper 2 active mode
- 0: Tamper 2 detection mode is passive.
- 1: Tamper 2 detection mode is active.

Bit 0 **TAMP1AM**: Tamper 1 active mode
- 0: Tamper 1 detection mode is passive.
- 1: Tamper 1 detection mode is active.

**Note:** Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

### 51.6.5 TAMP active tamper seed register (TAMP_ATSEEDR)

**Address offset:** 0x14

**Backup domain reset value:** 0x0000 0000

**System reset:** not affected

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

| SEED[15:0] |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

**Bits 31:0 SEED[31:0]:** Pseudo-random generator seed value

This register must be written four times with 32-bit values to provide the 128-bit seed to the PRNG. Writing to this register automatically sends the seed value to the PRNG.
51.6.6  **TAMP active tamper output register (TAMP_ATOR)**

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected, except for SEEDF which is reset to 0.

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 15</td>
<td><strong>INITS</strong>: Active tamper initialization status</td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware when the PRNG has absorbed the first 128-bit seed, meaning that the enabled active tampers are functional. This flag is left unchanged when the active tampers are disabled.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 14</th>
<th><strong>SEEDF</strong>: Seed running flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This flag is set by hardware when a new seed is written in the TAMP_ATSEEDR. It is cleared by hardware when the PRNG has absorbed this new seed, and by system reset. The TAMP APB cock must not be switched off as long as SEEDF is set.</td>
</tr>
</tbody>
</table>

| Bits 13:8  | Reserved, must be kept at reset value. |
| Bits 7:0   | **PRNG[7:0]**: Pseudo-random generator value |
|           | This field provides the values of the PRNG output. Because of potential inconsistencies due to synchronization delays, PRNG must be read at least twice. The read value is correct if it is equal to previous read value. |

51.6.7  **TAMP interrupt enable register (TAMP_IER)**

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected
RM0455 Tamper and backup registers (TAMP)

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 ITAMP8IE: Internal tamper 8 interrupt enable: monotonic counter overflow
   0: Internal tamper 8 interrupt disabled.
   1: Internal tamper 8 interrupt enabled.

Bit 22 Reserved, must be kept at reset value.

Bit 21 ITAMP6IE: Internal tamper 6 interrupt enable: ST manufacturer readout
   0: Internal tamper 6 interrupt disabled.
   1: Internal tamper 6 interrupt enabled.

Bit 20 ITAMP5IE: Internal tamper 5 interrupt enable: RTC calendar overflow
   0: Internal tamper 5 interrupt disabled.
   1: Internal tamper 5 interrupt enabled.

Bit 19 ITAMP4IE: Internal tamper 4 interrupt enable: HSE monitoring
   0: Internal tamper 4 interrupt disabled.
   1: Internal tamper 4 interrupt enabled.

Bit 18 ITAMP3IE: Internal tamper 3 interrupt enable: LSE monitoring
   0: Internal tamper 3 interrupt disabled.
   1: Internal tamper 3 interrupt enabled.

Bit 17 ITAMP2IE: Internal tamper 2 interrupt enable: Temperature monitoring
   0: Internal tamper 2 interrupt disabled.
   1: Internal tamper 2 interrupt enabled.

Bit 16 ITAMP1IE: Internal tamper 1 interrupt enable: RTC power domain supply monitoring
   0: Internal tamper 1 interrupt disabled.
   1: Internal tamper 1 interrupt enabled

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 TAMP3IE: Tamper 3 interrupt enable
   0: Tamper 3 interrupt disabled.
   1: Tamper 3 interrupt enabled.

Bit 1 TAMP2IE: Tamper 2 interrupt enable
   0: Tamper 2 interrupt disabled.
   1: Tamper 2 interrupt enabled.

Bit 0 TAMP1IE: Tamper 1 interrupt enable
   0: Tamper 1 interrupt disabled.
   1: Tamper 1 interrupt enabled.
51.6.8  **TAMP status register (TAMP_SR)**

Address offset: 0x30
Backup domain reset value: 0x0000 0000
System reset: not affected

![Register Table]

Bits 31:24  Reserved, must be kept at reset value.

- **Bit 23 ITAMP8F**: Monotonic counter overflow tamper flag
  - This flag is set by hardware when a tamper detection event is detected on the internal tamper 8.

- **Bit 22 Reserved, must be kept at reset value.**

- **Bit 21 ITAMP6F**: ST manufacturer readout tamper detection flag
  - This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

- **Bit 20 ITAMP5F**: RTC calendar overflow tamper detection flag
  - This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

- **Bit 19 ITAMP4F**: HSE monitoring tamper detection flag
  - This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.

- **Bit 18 ITAMP3F**: LSE monitoring tamper detection flag
  - This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

- **Bit 17 ITAMP2F**: Temperature monitoring tamper detection flag
  - This flag is set by hardware when a tamper detection event is detected on the internal tamper 2.

- **Bit 16 ITAMP1F**: RTC power domain voltage monitoring tamper detection flag
  - This flag is set by hardware when a tamper detection event is detected on the internal tamper 1.

Bits 15:3  Reserved, must be kept at reset value.

- **Bit 2 TAMP3F**: TAMP3 detection flag
  - This flag is set by hardware when a tamper detection event is detected on the TAMP3 input.

- **Bit 1 TAMP2F**: TAMP2 detection flag
  - This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.

- **Bit 0 TAMP1F**: TAMP1 detection flag
  - This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.
### 51.6.9 TAMP masked interrupt status register (TAMP_MISR)

Address offset: 0x34  
Backup domain reset value: 0x0000 0000  
System reset: not affected

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 23   | ITAMP8MF: Monotonic counter overflow interrupt masked flag  
|          | This flag is set by hardware when the internal tamper 8 interrupt is raised. |
| Bit 22   | Reserved, must be kept at reset value. |
| Bit 21   | ITAMP6MF: ST manufacturer readout tamper interrupt masked flag  
|          | This flag is set by hardware when the internal tamper 6 interrupt is raised. |
| Bit 20   | ITAMP5MF: RTC calendar overflow tamper interrupt masked flag  
|          | This flag is set by hardware when the internal tamper 5 interrupt is raised. |
| Bit 19   | ITAMP4MF: HSE monitoring tamper interrupt masked flag  
|          | This flag is set by hardware when the internal tamper 4 interrupt is raised. |
| Bit 18   | ITAMP3MF: LSE monitoring tamper interrupt masked flag  
|          | This flag is set by hardware when the internal tamper 3 interrupt is raised. |
| Bit 17   | ITAMP2MF: Temperature monitoring tamper interrupt masked flag  
|          | This flag is set by hardware when the internal tamper 2 interrupt is raised. |
| Bit 16   | ITAMP1MF: RTC power domain voltage monitoring tamper interrupt masked flag  
|          | This flag is set by hardware when the internal tamper 1 interrupt is raised. |

Bits 15:3 Reserved, must be kept at reset value.

| Bit 2   | TAMP3MF: TAMP3 interrupt masked flag  
|         | This flag is set by hardware when the tamper 3 interrupt is raised. |
| Bit 1   | TAMP2MF: TAMP2 interrupt masked flag  
|         | This flag is set by hardware when the tamper 2 interrupt is raised. |
| Bit 0   | TAMP1MF: TAMP1 interrupt masked flag  
|         | This flag is set by hardware when the tamper 1 interrupt is raised. |
51.6.10  **TAMP status clear register (TAMP_SCR)**

Address offset: 0x3C

System reset value: 0x0000 0000

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

Bits 31:24  Reserved, must be kept at reset value.

Bit 23  **CITAMP8F**: Clear ITAMP8 detection flag
Writing 1 in this bit clears the ITAMP8F bit in the TAMP_SR register.

Bit 22  Reserved, must be kept at reset value.

Bit 21  **CITAMP6F**: Clear ITAMP6 detection flag
Writing 1 in this bit clears the ITAMP6F bit in the TAMP_SR register.

Bit 20  **CITAMP5F**: Clear ITAMP5 detection flag
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19  **CITAMP4F**: Clear ITAMP4 detection flag
Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.

Bit 18  **CITAMP3F**: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17  **CITAMP2F**: Clear ITAMP2 detection flag
Writing 1 in this bit clears the ITAMP2F bit in the TAMP_SR register.

Bit 16  **CITAMP1F**: Clear ITAMP1 detection flag
Writing 1 in this bit clears the ITAMP1F bit in the TAMP_SR register.

Bits 15:3  Reserved, must be kept at reset value.

Bit 2  **CTAMP3F**: Clear TAMP3 detection flag
Writing 1 in this bit clears the TAMP3F bit in the TAMP_SR register.

Bit 1  **CTAMP2F**: Clear TAMP2 detection flag
Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.

Bit 0  **CTAMP1F**: Clear TAMP1 detection flag
Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.
### 51.6.11 TAMP monotonic counter register (TAMP_COUNTR)

Address offset: 0x040  
Backup domain reset value: 0x0000 0000  
System reset: not affected

<table>
<thead>
<tr>
<th>Address</th>
<th>Format</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-16</td>
<td>COUNT[31:16]</td>
<td>This register is read-only only and is incremented by one when a write access is done to this register. This register cannot roll-over and is frozen when reaching the maximum value.</td>
<td>r r r r r r r r r r r r r r r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Format</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-0</td>
<td>COUNT[15:0]</td>
<td>Bits 31:0 COUNT[31:0]</td>
<td>r r r r r r r r r r r r r r r</td>
</tr>
</tbody>
</table>

### 51.6.12 TAMP configuration register (TAMP_CFG)

Address offset: 0x050  
Backup domain reset value: 0x0000 0000  
System reset: not affected

<table>
<thead>
<tr>
<th>Address</th>
<th>Format</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Format</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Format</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
</table>
| 0       | OUT3_RMP: TAMP_OUT3 mapping | 0x0: TAMP_OUT3 is mapped on PI8  
0x1: TAMP_OUT3 is mapped on PC13 | 0x0: TAMP_OUT3 is mapped on PI8  
0x1: TAMP_OUT3 is mapped on PC13 |
51.6.13  TAMP backup x register (TAMP_BKPxR)

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)
Backup domain reset value: 0x0000 0000
System reset: not affected

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

Bits 31:0  BKP[31:0]

The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so that they are not reset by System reset, and their contents remain valid when the device operates in low-power mode.

In the default configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.
# TAMP register map

## Table 361. TAMP register map and reset values

| Offset | Register       | 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   | 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |
|--------|----------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 0x00   | TAMP_CR1       |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Reset value |            | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    |
| 0x04   | TAMP_CR2       |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Reset value |            | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
| 0x0C   | TAMP_FLTCR     |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Reset value |            | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
| 0x10   | TAMP_ATCR1     |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Reset value |            | 0    | 0    | 0    | 0    | 0    | 1    | 1    | 1    | 1    | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
| 0x14   | TAMP_ATSEEDR   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Reset value |            | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
| 0x18   | TAMP_ATOR      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Reset value |            | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
| 0x2C   | TAMP_IER       |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Reset value |            | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
| 0x30   | TAMP_SR        |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Reset value |            | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
| 0x34   | TAMP_MISR      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Reset value |            | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
| 0x3C   | TAMP_SCR       |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Reset value |            | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
| 0x40   | TAMP_COUNTR    |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Reset value |            | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
Refer to Section 2.3 on page 128 for the register boundary addresses.
52 Inter-integrated circuit (I2C) interface

52.1 Introduction

The I²C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I²C bus. It provides multimaster capability, and controls all I²C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

It is also SMBus (system management bus) and PMBus (power management bus) compatible.

DMA can be used to reduce CPU overload.

52.2 I2C main features

- I²C bus specification rev03 compatibility:
  - Slave and master modes
  - Multimaster capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
The following additional features are also available depending on the product implementation (see Section 52.3: I2C implementation):

- SMBus specification rev 3.0 compatibility:
  - Hardware PEC (Packet Error Checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and Device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock: a choice of independent clock sources allowing the I2C communication speed to be independent from the i2c_pclk reprogramming
- Wakeup from Stop mode on address match.

### 52.3 I2C implementation

This manual describes the full set of features implemented in I2C peripheral. In the STM32H7xxx devices I2C1, I2C2, I2C3 and I2C4 implement the full set of features as shown in the following table.

#### Table 362. STM32H7A3/7B3 and STM32H7B0 I2C implementation

<table>
<thead>
<tr>
<th>I2C features(1)</th>
<th>I2C1</th>
<th>I2C2</th>
<th>I2C3</th>
<th>I2C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-bit addressing mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>10-bit addressing mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Standard-mode (up to 100 kbit/s)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Fast-mode (up to 400 kbit/s)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Fast-mode Plus with 20mA output drive I/Os (up to 1 Mbit/s)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Independent clock</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Wakeup from Stop mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SMBus/PMBus</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

1. X = supported.

### 52.4 I2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I²C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I²C bus.

This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).
If SMBus feature is supported: the additional optional SMBus Alert pin (SMBA) is also available.

52.4.1 I2C block diagram

The block diagram of the I2C interface is shown in Figure 505.

![Figure 505. I2C block diagram](image)

The I2C is clocked by an independent clock source which allows the I2C to operate independently from the i2c_pclk frequency.

For I2C I/Os supporting 20 mA output current drive for Fast-mode Plus operation, the driving capability is enabled through control bits in the system configuration controller (SYSCFG).

Refer to Section 52.3: I2C implementation.
52.4.2 I2C pins and internal signals

Table 363. I2C input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C_SDA</td>
<td>Bidirectional</td>
<td>I2C data</td>
</tr>
<tr>
<td>I2C_SCL</td>
<td>Bidirectional</td>
<td>I2C clock</td>
</tr>
<tr>
<td>I2C_SMBA</td>
<td>Bidirectional</td>
<td>SMBus Alert</td>
</tr>
</tbody>
</table>

Table 364. I2C internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i2c_ker_ck</td>
<td>Input</td>
<td>I2C kernel clock, also named I2CCLK in this document</td>
</tr>
<tr>
<td>i2c_pclk</td>
<td>Input</td>
<td>I2C APB clock</td>
</tr>
<tr>
<td>i2c_it</td>
<td>Output</td>
<td>I2C interrupts, refer to Table 378: I2C Interrupt requests for the full list of interrupt sources</td>
</tr>
<tr>
<td>i2c_rx_dma</td>
<td>Output</td>
<td>I2C Receive Data DMA request (I2C_RX)</td>
</tr>
<tr>
<td>i2c_tx_dma</td>
<td>Output</td>
<td>I2C Transmit Data DMA request (I2C_TX)</td>
</tr>
</tbody>
</table>

52.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period \( t_{I2CCLK} \) must respect the following conditions:

\[
t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4 \quad \text{and} \quad t_{I2CCLK} < t_{HIGH}
\]

with:

- \( t_{LOW} \): SCL low time and \( t_{HIGH} \): SCL high time
- \( t_{filters} \): when enabled, sum of the delays brought by the analog filter and by the digital filter.

Analog filter delay is maximum 260 ns. Digital filter delay is \( DNF \times t_{I2CCLK} \).

The i2c_pclk clock period \( t_{PCLK} \) must respect the following condition:

\[
t_{PCLK} < 4/3 \ t_{SCL}
\]

with \( t_{SCL} \): SCL period

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for \( t_{I2CCLK} \).

52.4.4 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver
By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

**Communication flow**

In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.

![Figure 506. I2C bus protocol](MS19854V1)

Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

### 52.4.5 I2C initialization

#### Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller.

Then the I2C can be enabled by setting the PE bit in the I2C\_CR1 register.

When the I2C is disabled (PE=0), the I\(^2\)C performs a software reset. Refer to Section 52.4.6: Software reset for more details.

#### Noise filters

Before enabling the I2C peripheral by setting the PE bit in I2C\_CR1 register, the user must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I\(^2\)C specification which requires the
suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. The user can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x i2c_ker_ck periods. This allows spikes with a programmable length of 1 to 15 i2c_ker_ck periods to be suppressed.

Table 365. Comparison of analog vs. digital filters

<table>
<thead>
<tr>
<th></th>
<th>Analog filter</th>
<th>Digital filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse width of</td>
<td>≥ 50 ns</td>
<td>Programmable length from 1 to 15 I2C peripheral</td>
</tr>
<tr>
<td>suppressed spikes</td>
<td></td>
<td>clocks</td>
</tr>
<tr>
<td>Benefits</td>
<td>Available in Stop mode</td>
<td>Programmable length: extra filtering capability vs. standard requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-- Stable length</td>
</tr>
<tr>
<td>Drawbacks</td>
<td>Variation vs. temperature, voltage, process</td>
<td>Wakeup from Stop mode on address match is not available when digital filter is enabled</td>
</tr>
</tbody>
</table>

**Caution:** Changing the filter configuration is not allowed when the I2C is enabled.
I2C timings

The timings must be configured in order to guarantee a correct data hold and setup time, used in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C configuration window.

**Figure 507. Setup and hold timings**

Data hold time: in case of transmission, the data is sent on SDA output after the SDADEL delay, if it is already available in I2C_TXDR.

Data setup time: in case of transmission, the SCLDEL counter starts when the data is sent on SDA output.
When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is \( t_{SDADEL} = SDADEL \times t_{PRESC} + t_{I2CCLK} \) where \( t_{PRESC} = (PRESC+1) \times t_{I2CCLK} \).  

\( t_{SDADEL} \) impacts the hold time \( t_{HD;DAT} \).

The total SDA output delay is:

\[
t_{SYNC1} + \left\{ SDADEL \times (PRESC+1) + 1 \right\} \times t_{I2CCLK}
\]

\( t_{SYNC1} \) duration depends on these parameters:

- SCL falling slope
- When enabled, input delay brought by the analog filter: \( t_{AF(min)} < t_{AF} < t_{AF(max)} \)
- When enabled, input delay brought by the digital filter: \( t_{DNF} = DNF \times t_{I2CCLK} \)
- Delay due to SCL synchronization to i2c_ker_ck clock (2 to 3 i2c_ker_ck periods)

In order to bridge the undefined region of the SCL falling edge, the user must program SDADEL in such a way that:

\[
\left\{ t_{(max)} + t_{HD;DAT(min)} - t_{AF(min)} - [(DNF+3) \times t_{I2CCLK}] \right\} / \left\{ (PRESC+1) \times t_{I2CCLK} \right\} \leq SDADEL
\]

\[
SDADEL \leq \left\{ t_{VD;DAT(max)} - t_{AF(max)} - [(DNF+4) \times t_{I2CCLK}] \right\} / \left\{ (PRESC+1) \times t_{I2CCLK} \right\}
\]

**Note:** \( t_{AF(min)}/t_{AF(max)} \) are part of the equation only when the analog filter is enabled. Refer to device datasheet for \( t_{AF} \) values.

The maximum \( t_{VD;DAT} \) can be 3.45 µs, 0.9 µs and 0.45 µs for Standard-mode, Fast-mode and Fast-mode Plus, but must be less than the maximum of \( t_{VD;DAT} \) by a transition time. This maximum must only be met if the device does not stretch the LOW period (\( t_{LOW} \)) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case, so in this case the previous equation becomes:

\[
SDADEL \leq \left\{ t_{VD;DAT(max)} - t_{(max)} - 260 \text{ ns} - [(DNF+4) \times t_{I2CCLK}] \right\} / \left\{ (PRESC+1) \times t_{I2CCLK} \right\}
\]

**Note:** This condition can be violated when \( NOSTRETCH=0 \), because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL value.

Refer to Table 366: I2C-SMBus specification data setup and hold times for \( t_{r} \), \( t_{t} \), \( t_{HD;DAT} \) and \( t_{VD;DAT} \) standard values.

After \( t_{SDADEL} \) delay, or after sending SDA output in case the slave had to stretch the clock because the data was not yet written in I2C_TXDR register, SCL line is kept at low level during the setup time. This setup time is \( t_{SCLDEL} = (SCLDEL+1) \times t_{PRESC} \) where \( t_{PRESC} = (PRESC+1) \times t_{I2CCLK} \).

\( t_{SCLDEL} \) impacts the setup time \( t_{SU;DAT} \).

In order to bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL in such a way that:

\[
\left\{ t_{r(max)} + t_{SU;DAT(min)} \right\} / \left\{ (PRESC+1) \times t_{I2CCLK} \right\} - 1 \leq SCLDEL
\]

Refer to Table 366: I2C-SMBus specification data setup and hold times for \( t_{r} \) and \( t_{SU;DAT} \) standard values.

The SDA and SCL transition time values to be used are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature whatever the application.
At every clock pulse, after SCL falling edge detection, the I2C master or slave stretches SCL low during at least \([(\text{SDADEL} + \text{SCLDEL} + 1) \times (\text{PRESC} + 1) + 1] \times t_{\text{I2CCLK}}\) in both transmission and reception modes. In transmission mode, in case the data is not yet written in I2C_TXDR when SDADEL counter is finished, the I2C keeps on stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

If NOSTRETCH=1 in slave mode, the SCL is not stretched. Consequently the SDADEL must be programmed in such a way to guarantee also a sufficient setup time.

Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is \(t_{\text{SCLL}} = (\text{SCLL} + 1) \times t_{\text{PRESC}}\) where \(t_{\text{PRESC}} = (\text{PRESC} + 1) \times t_{\text{I2CCLK}}\). \(t_{\text{SCLL}}\) impacts the SCL low time \(t_{\text{LOW}}\).

- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is \(t_{\text{SCLH}} = (\text{SCLH} + 1) \times t_{\text{PRESC}}\) where \(t_{\text{PRESC}} = (\text{PRESC} + 1) \times t_{\text{I2CCLK}}\). \(t_{\text{SCLH}}\) impacts the SCL high time \(t_{\text{HIGH}}\).

Refer to \textit{I2C master initialization} for more details.

**Caution:** Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral. Refer to \textit{I2C slave initialization} for more details.

**Caution:** Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.
52.4.6 Software reset

A software reset can be performed by clearing the PE bit in the I2C_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:
1. I2C_CR2 register: START, STOP, NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:
1. I2C_CR2 register: PECBYTE
2. I2C_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least 3 APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence: - Write PE=0 - Check PE=0 - Write PE=1.
52.4.7 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the 8th SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE=0). If RXNE=1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the 8th and 9th SCL pulse (before the Acknowledge pulse).

Figure 509. Data reception
Transmission

If the I2C_TXDR register is not empty (TXE=0), its content is copied into the shift register after the 9th SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE=1, meaning that no data is written yet in I2C_TXDR, SCL line is stretched low until I2C_TXDR is written. The stretch is done after the 9th SCL pulse.

Figure 510. Data transmission

Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in master mode. By default it is disabled in slave mode, but it can be enabled by software by setting the SBC (Slave Byte Control) bit in the I2C_CR2 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this mode, TCR flag is set when the number of bytes programmed in NBYTES has been transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.
When RELOAD=0 in master mode, the counter can be used in 2 modes:

- **Automatic end mode** (AUTOEND = ‘1’ in the I2C_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred.

- **Software end mode** (AUTOEND = ‘0’ in the I2C_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2C_CR2 register. This mode must be used when the master wants to send a RESTART condition.

**Caution:** The AUTOEND bit has no effect when the RELOAD bit is set.

### Table 367. I2C configuration

<table>
<thead>
<tr>
<th>Function</th>
<th>SBC bit</th>
<th>RELOAD bit</th>
<th>AUTOEND bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Tx/Rx NBYTES + STOP</td>
<td>x</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Master Tx/Rx + NBYTES + RESTART</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Slave Tx/Rx</td>
<td>0</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>all received bytes ACKed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slave Rx with ACK control</td>
<td>1</td>
<td>1</td>
<td>x</td>
</tr>
</tbody>
</table>

### 52.4.8 I2C slave mode

#### I2C slave initialization

In order to work in slave mode, the user must enable at least one slave address. Two registers I2C_OAR1 and I2C_OAR2 are available in order to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C_OAR1 register.
  OA1 is enabled by setting the OA1EN bit in the I2C_OAR1 register.

- If additional slave addresses are required, the 2nd slave address OA2 can be configured. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK=7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.
  These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK=0.
  OA2 is enabled by setting the OA2EN bit in the I2C_OAR2 register.

- The General Call address is enabled by setting the GCEN bit in the I2C_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.
By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, in order to perform software actions. If the master does not support clock stretching, the I2C must be configured with NOSTRETCH=1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled the user must read the ADDCODE[6:0] bits in the I2C_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

**Slave clock stretching (NOSTRETCH = 0)**

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C_TXDR register, or if the first data byte is not written when the ADDR flag is cleared (TXE=1). This stretch is released when the data is written to the I2C_TXDR register.
- In reception when the I2C_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C_RXDR is read.
- When TCR = 1 in Slave Byte Control mode, reload mode (SBC=1 and RELOAD=1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during 
\[(SDADEL+SCLDEL+1) \times (PRESC+1) + 1\] \times t_{I2CCLK}.

**Slave without clock stretching (NOSTRETCH = 1)**

When NOSTRETCH = 1 in the I2C_CR1 register, the I2C slave does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, he ensures that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2C_RXDR register before the 9th SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.
Slave Byte Control mode

In order to allow byte ACK control in slave reception mode, Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. This is required to be compliant with SMBus standards.

Reload mode must be selected in order to allow byte ACK control in slave reception mode (RELOAD=1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the 8th and 9th SCL pulses. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

Note: The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR=1.

The RELOAD bit value can be changed when ADDR=1, or when TCR=1.

Caution: Slave Byte Control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH=1 is not allowed.

Figure 511. Slave initialization flowchart

---

**Figure 511. Slave initialization flowchart**

- Slave initialization
- Initial settings
- Clear {OA1EN, OA2EN} in I2C_OAR1 and I2C_OAR2
- Configure {OA1[9:0], OA1MODE, OA1EN, OA2[6:0], OA2MSK[2:0], OA2EN, GCEN}
- Configure SBC in I2C_CR1*
- Enable interrupts and/or DMA in I2C_CR1
- End

*SBC must be set to support SMBus features
Slave transmitter

A transmit interrupt status (TXIS) is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C_CR1 register.

The TXIS bit is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C_CR1 register. The slave automatically releases the SCL and SDA lines in order to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C_CR1 register, the STOPF flag is set in the I2C_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to ‘0’. In this case, if TXE = 0 when the slave address is received (ADDR=1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register by setting the TXE bit in order to program a new data byte.

In Slave Byte Control mode (SBC=1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR=1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

Caution: When NOSTRETCH=1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine, in order to program the first data byte. The first data byte to be sent must be previously programmed in the I2C_TXDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C_TXDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).
If a TXIS event is needed, (Transmit Interrupt or Transmit DMA request), the user must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.
Figure 512. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH= 0

- Slave transmission
  - Slave initialization
  - \( \text{i2c_isr.addr} = 1? \)
    - No
      - Read ADDCODE and DIR in I2C_ISR
        - Optional: Set I2C_ISR.TXE = 1
        - Set I2C_ICR.ADDRCF
    - Yes
      - SCL stretched

- \( \text{i2c_isr.txis} = 1? \)
  - No
  - Yes
    - Write I2C_TXDR.TXDATA
Figure 513. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH= 1

Slave transmission

Slave initialization

I2C_ISR.TXIS =1?

Yes

Write I2C_TXDR.TXDATA

No

I2C_ISR.STOPF =1?

Yes

Optional: Set I2C_ISR.TXE = 1 and I2C_ISR.TXIS=1

No

Set I2C_ICR.STOPCF
Figure 514. Transfer bus diagrams for I2C slave transmitter

Example I2C slave transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:

Example I2C slave transmitter 3 bytes without 1st data flush, NOSTRETCH=0:

Example I2C slave transmitter 3 bytes, NOSTRETCH=1:

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)
Slave receiver

RXNE is set in I2C_ISR when the I2C_RXDR is full, and generates an interrupt if RXIE is set in I2C_CR1. RXNE is cleared when I2C_RXDR is read.

When a STOP is received and STOPIE is set in I2C_CR1, STOPF is set in I2C_ISR and an interrupt is generated.

Figure 515. Transfer sequence flowchart for slave receiver with NOSTRETCH=0
Figure 516. Transfer sequence flowchart for slave receiver with NOSTRETCH=1

Slave reception

Slave initialization

I2C_ISR.RXNE = 1?
Yes
Read I2C_RXDR.RXDATA

No
Set I2C_ICR.STOPCF

I2C_ISR.STOPF = 1?
Yes

No

Figure 517. Transfer bus diagrams for I2C slave receiver

Example I2C slave receiver 3 bytes, NOSTRETCH=0:

S Address A
EV1

RXNE

A data1 A data2

RXNE

A data3 A

RXNE

EV2

EV3

EV4

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: RXNE ISR: rd data1
EV3: RXNE ISR: rd data2
EV4: RXNE ISR: rd data3

Example I2C slave receiver 3 bytes, NOSTRETCH=1:

S Address A
EV1

RXNE

A data 1 A data 2

RXNE

A data 3 A

RXNE

EV2

EV3

EV4

EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2
EV3: RXNE ISR: rd data3
EV4: STOPF ISR: set STOPCF
52.4.9 I2C master mode

I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

A clock synchronization mechanism is implemented in order to support multi-master environment and slave clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a tSYNC1 delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases SCL to high level once the SCCL counter reaches the value programmed in the SCCL[7:0] bits in the I2C_TIMINGR register.

The I2C detects its own SCL high level after a tSYNC2 delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCCH counter is reached reaches the value programmed in the SCCH[7:0] bits in the I2C_TIMINGR register.

Consequently the master clock period is:

\[ t_{SCL} = t_{SYNC1} + t_{SYNC2} + \left( \left( SCLH+1 \right) + \left( SCLL+1 \right) \right) \times \text{(PRESC+1)} \times t_{I2CCLK} \]

The duration of \( t_{SYNC1} \) depends on these parameters:
- SCL falling slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF \( \times t_{I2CCLK} \)
- Delay due to SCL synchronization with \( i2c\_ker\_ck \) clock (2 to 3 \( i2c\_ker\_ck \) periods)

The duration of \( t_{SYNC2} \) depends on these parameters:
- SCL rising slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF \( \times t_{I2CCLK} \)
- Delay due to SCL synchronization with \( i2c\_ker\_ck \) clock (2 to 3 \( i2c\_ker\_ck \) periods)
Caution: In order to be \(\text{i}^2\text{C}\) or SMBus compliant, the master clock must respect the timings given below:
**Master communication initialization (address phase)**

In order to initiate the communication, the user must program the following parameters for the addressed slave in the I2C_CR2 register:

- **Addressing mode (7-bit or 10-bit): ADD10**
- **Slave address to be sent: SADD[9:0]**
- **Transfer direction: RD_WRN**
- **In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.**
- **The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.**

The user must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of tBUF.

In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

**Note:** The START bit is reset by hardware when the slave address has been sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs.

**In 10-bit addressing mode, when the Slave Address first 7 bits is NACKed by the slave,** the

---

**Table 368. I²C-SMBus specification clock timings**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
<th>SMBus</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>fSCL</td>
<td>SCL clock frequency</td>
<td>-</td>
<td>100</td>
<td>-</td>
<td>400</td>
<td>-</td>
</tr>
<tr>
<td>tHD:STA</td>
<td>Hold time (repeated) START condition</td>
<td>4.0</td>
<td>-</td>
<td>0.6</td>
<td>-</td>
<td>0.26</td>
</tr>
<tr>
<td>tSU:STA</td>
<td>Set-up time for a repeated START condition</td>
<td>4.7</td>
<td>-</td>
<td>0.6</td>
<td>-</td>
<td>0.26</td>
</tr>
<tr>
<td>tSU:STO</td>
<td>Set-up time for STOP condition</td>
<td>4.0</td>
<td>-</td>
<td>0.6</td>
<td>-</td>
<td>0.26</td>
</tr>
<tr>
<td>tBUF</td>
<td>Bus free time between a STOP and START</td>
<td>4.7</td>
<td>-</td>
<td>1.3</td>
<td>-</td>
<td>0.5</td>
</tr>
<tr>
<td>tLOW</td>
<td>Low period of the SCL clock</td>
<td>4.7</td>
<td>-</td>
<td>1.3</td>
<td>-</td>
<td>0.5</td>
</tr>
<tr>
<td>tHIGH</td>
<td>Period of the SCL clock</td>
<td>4.0</td>
<td>-</td>
<td>0.6</td>
<td>-</td>
<td>0.26</td>
</tr>
<tr>
<td>tR</td>
<td>Rise time of both SDA and SCL signals</td>
<td>-</td>
<td>1000</td>
<td>-</td>
<td>300</td>
<td>-</td>
</tr>
<tr>
<td>tF</td>
<td>Fall time of both SDA and SCL signals</td>
<td>-</td>
<td>300</td>
<td>-</td>
<td>300</td>
<td>-</td>
</tr>
</tbody>
</table>

**Note:**

- **SCLL is also used to generate the tBUF and tSU:STA timings.**
- **SCLH is also used to generate the tHD:STA and tSU:STO timings.**

Refer to [Section 52.4.10: I2C_TIMINGR register configuration examples](#) for examples of I2C_TIMINGR settings vs. i2c_ker_ck frequency.
master re-launches automatically the slave address transmission until ACK is received. In this case ADDRCF must be set if a NACK is received from the slave, in order to stop sending the slave address.

If the I2C is addressed as a slave (ADDR=1) while the START bit is set, the I2C switches to slave mode and the START bit is cleared, when the ADDRCF bit is set.

Note: The same procedure is applied for a Repeated Start condition. In this case BUSY=1.

Initialization of a master receiver addressing a 10-bit address slave

- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set: (Re)Start + Slave address 10-bit header Write + Slave address 2nd byte + REStart + Slave address 10-bit header Read
• If the master addresses a 10-bit address slave, transmits data to this slave and then reads data from the same slave, a master transmission flow must be done first. Then a repeated start is set with the 10 bit slave address configured with HEAD10R=1. In this case the master sends this sequence: ReStart + Slave address 10-bit header Read.

**Figure 521. 10-bit address read access with HEAD10R=1**

Master transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the 9th SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C_CR1 register. The flag is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

• When RELOAD=0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND=1), a STOP is automatically sent.
  - In software end mode (AUTOEND=0), the TC flag is set and the SCL line is stretched low in order to perform software actions:
    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.
    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

• If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2C_ISR register, and an interrupt is generated if the NACKIE bit is set.
Figure 522. Transfer sequence flowchart for I2C master transmitter for N≤255 bytes

Master transmission

Master initialization

NBYTES = N
AUTOEND = 0 for RESTART, 1 for STOP
Configure slave address
Set I2C_CR2.START

No

I2C_ISR.NACKF = 1?
Yes
End

I2C_ISR.TXIS = 1?
No

Yes

Write I2C_TXDR

NBYTES transmitted?
Yes

I2C_ISR.TC = 1?
No

Set I2C_CR2.START with slave address NBYTES ...

No
Figure 523. Transfer sequence flowchart for I2C master transmitter for N>255 bytes

Master initialization

NBYTES = 0xFF; N=N-255
RELOAD = 1
Configure slave address
Set I2C_CR2.START

I2C_ISR.NACKF = 1?
Yes
End

No

I2C_ISR.TXIS = 1?
Yes
Write I2C_TXDR

No

I2C_ISR.TXIS = 1?
Yes

No

IF N< 256
NBYTES = N; N = 0; RELOAD = 0
AUTOEND = 0 for RESTART; 1 for STOP
ELSE
NBYTES = 0xFF; N = N-255
RELOAD = 1

NBYTES transmitted ?
Yes

IF N= 256

Set I2C_CR2.START with slave address
NBYTES ...

No

I2C_ISR.TCR = 1?
Yes

END

No

I2C_ISR.TC = 1?
Yes

No

No

No
**Figure 524. Transfer bus diagrams for I2C master transmitter**

### Example I2C master transmitter 2 bytes, automatic end mode (STOP)

- **INIT**: program Slave address, program NBYTES = 2, AUTOEND=1, set START
- **EV1**: TXIS ISR: wr data1
- **EV2**: TXIS ISR: wr data2

![Transfer bus diagram for I2C master transmitter (STOP)](image)

### Example I2C master transmitter 2 bytes, software end mode (RESTART)

- **INIT**: program Slave address, program NBYTES = 2, AUTOEND=0, set START
- **EV1**: TXIS ISR: wr data1
- **EV2**: TXIS ISR: wr data2
- **EV3**: TC ISR: program Slave address, program NBYTES = N, set START

![Transfer bus diagram for I2C master transmitter (RESTART)](image)
Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the 8th SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C_CR1 register. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD=0 and NBYTES[7:0] data have been transferred:
  - In automatic end mode (AUTOEND=1), a NACK and a STOP are automatically sent after the last received byte.
  - In software end mode (AUTOEND=0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:
    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.
    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
Figure 525. Transfer sequence flowchart for I2C master receiver for N≤255 bytes

Master reception

Master initialization

NBYTES = N
AUTOEND = 0 for RESTART; 1 for STOP
Configure slave address
Set I2C_CR2.START

I2C_ISR.RXNE = 1?

Yes

Read I2C_RXDR

NBYTES received?

Yes

Set I2C_CR2.START with slave address NBYTES...

No

End

No

I2C_ISR.TC = 1?

Yes

No
Figure 526. Transfer sequence flowchart for I2C master receiver for N >255 bytes

Master initialization

NBYTES = 0xFF; N=N-255
RELOAD = 1
Configure slave address
Set I2C_CR2.START

I2C_ISR.RXNE =1?

Yes

Read I2C_RXDR

NBYTES received?

Yes

IF N< 256
NBYTES =N; N=0;RELOAD=0
AUTOEND=0 for RESTART; 1 for STOP
ELSE
NBYTES =0xFF;N=N-255
RELOAD=1

No

No

Set I2C_CR2.START with slave address NBYTES ...

End

I2C_ISR.TC = 1?

No

I2C_ISR.TCR = 1?

No

Yes

Yes

No
Example I2C master receiver 2 bytes, automatic end mode (STOP)

INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2

Example I2C master receiver 2 bytes, software end mode (RESTART)

INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: read data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START
### 52.4.10 I2C_TIMINGR register configuration examples

The tables below provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I2C specification. In order to get more accurate configuration values, the STM32CubeMX tool (I2C Configuration window) must be used.

#### Table 369. Examples of timing settings for $f_{I2CCLK} = 8$ MHz

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 kHz</td>
<td>100 kHz</td>
<td>400 kHz</td>
</tr>
<tr>
<td>PRESC</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SCLL</td>
<td>0xC7</td>
<td>0x13</td>
<td>0x9</td>
</tr>
<tr>
<td>tSCLL</td>
<td>200x250 ns = 50 µs</td>
<td>20x250 ns = 5.0 µs</td>
<td>10x125 ns = 1250 ns</td>
</tr>
<tr>
<td>SCLH</td>
<td>0xC3</td>
<td>0xF</td>
<td>0x3</td>
</tr>
<tr>
<td>tSCLH</td>
<td>196x250 ns = 49 µs</td>
<td>16x250 ns = 4.0 µs</td>
<td>4x125 ns = 500 ns</td>
</tr>
<tr>
<td>tSCL</td>
<td>~100 µs$^{(2)}$</td>
<td>~10 µs$^{(2)}$</td>
<td>~2500 ns$^{(3)}$</td>
</tr>
<tr>
<td>SDADEL</td>
<td>0x2</td>
<td>0x2</td>
<td>0x1</td>
</tr>
<tr>
<td>tSDADEL</td>
<td>2x250 ns = 500 ns</td>
<td>2x250 ns = 500 ns</td>
<td>1x125 ns = 125 ns</td>
</tr>
<tr>
<td>SCLDEL</td>
<td>0x4</td>
<td>0x4</td>
<td>0x3</td>
</tr>
<tr>
<td>tSCLDEL</td>
<td>5x250 ns = 1250 ns</td>
<td>5x250 ns = 1250 ns</td>
<td>4x125 ns = 500 ns</td>
</tr>
</tbody>
</table>

1. $SCL$ period $t_{SCL}$ is greater than $t_{SCLL} + t_{SCLH}$ due to SCL internal detection delay. Values provided for $t_{SCL}$ are examples only.
2. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 500$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 1000$ ns.
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 500$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 750$ ns.
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 500$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 655$ ns.

#### Table 370. Examples of timing settings for $f_{I2CCLK} = 16$ MHz

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 kHz</td>
<td>100 kHz</td>
<td>400 kHz</td>
</tr>
<tr>
<td>PRESC</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SCLL</td>
<td>0xC7</td>
<td>0x13</td>
<td>0x9</td>
</tr>
<tr>
<td>tSCLL</td>
<td>200 x 250 ns = 50 µs</td>
<td>20 x 250 ns = 5.0 µs</td>
<td>10 x 125 ns = 1250 ns</td>
</tr>
<tr>
<td>SCLH</td>
<td>0xC3</td>
<td>0xF</td>
<td>0x3</td>
</tr>
<tr>
<td>tSCLH</td>
<td>196 x 250 ns = 49 µs</td>
<td>16 x 250 ns = 4.0 µs</td>
<td>4 x 125 ns = 500 ns</td>
</tr>
<tr>
<td>tSCL</td>
<td>~100 µs$^{(2)}$</td>
<td>~10 µs$^{(2)}$</td>
<td>~2500 ns$^{(3)}$</td>
</tr>
<tr>
<td>SDADEL</td>
<td>0x2</td>
<td>0x2</td>
<td>0x2</td>
</tr>
<tr>
<td>tSDADEL</td>
<td>2 x 250 ns = 500 ns</td>
<td>2 x 250 ns = 500 ns</td>
<td>2 x 125 ns = 250 ns</td>
</tr>
<tr>
<td>SCLDEL</td>
<td>0x4</td>
<td>0x4</td>
<td>0x3</td>
</tr>
<tr>
<td>tSCLDEL</td>
<td>5 x 250 ns = 1250 ns</td>
<td>5 x 250 ns = 1250 ns</td>
<td>4 x 125 ns = 500 ns</td>
</tr>
</tbody>
</table>

1. $SCL$ period $t_{SCL}$ is greater than $t_{SCLL} + t_{SCLH}$ due to SCL internal detection delay. Values provided for $t_{SCL}$ are examples only.
52.4.11 SMBus specific features

This section is relevant only when SMBus feature is supported. Refer to Section 52.3: I2C implementation.

Introduction

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I²C principles of operation. SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBus specification (http://smbus.org).

The System Management Bus Specification refers to three types of devices.

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system’s CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as master or slave device, and also as a host.
Bus protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Read Byte, Read Word, Process Call, Block Read, Block Write and Block Write-Block Read Process Call. These protocols should be implemented by the user software.

For more details of these protocols, refer to SMBus specification (http://smbus.org).

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C_CR1 register. The ARP commands should be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus Address Resolution Protocol, refer to SMBus specification (http://smbus.org).

Received Command and Data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2C_CR1 register. Refer to Slave Byte Control mode on page 1895 for more details.

Host Notify protocol

This peripheral supports the Host Notify protocol by setting the SMBHEN bit in the I2C_CR1 register. In this case the host acknowledges the SMBus Host Address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

SMBus alert

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the Alert Response Address (0b0001 100). Only the device(s) which pulled SMBALERT# low acknowledges the Alert Response Address.

When configured as a slave device(SMBHEN=0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN=1), the ALERT flag is set in the I2C_ISR register when a falling edge is detected on the SMBA pin and ALERTEN=1. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. When ALERTEN=0, the ALERT line is considered high even if the external SMBA pin is low.

If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN=0.
Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. Packet Error Checking is implemented by appending a Packet Error Code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows a Not Acknowledge to be sent automatically when the received byte does not match with the hardware calculated PEC.

Timeouts

This peripheral embeds hardware timers in order to be compliant with the 3 timeouts defined in SMBus specification.

<table>
<thead>
<tr>
<th>Table 372. SMBus timeout specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>$t_{\text{TIMEOUT}}$</td>
</tr>
<tr>
<td>$t_{\text{LOW:SEXT}}$</td>
</tr>
<tr>
<td>$t_{\text{LOW:MEXT}}$</td>
</tr>
</tbody>
</table>

1. $t_{\text{LOW:SEXT}}$ is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that, another slave device or the master also extends the clock causing the combined clock low extend time to be greater than $t_{\text{LOW:SEXT}}$. Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.

2. $t_{\text{LOW:MEXT}}$ is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master also extends the clock causing the combined clock low time to be greater than $t_{\text{LOW:MEXT}}$ on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.
Figure 528. Timeout intervals for $t_{\text{LOW}:\text{SEXT}}$, $t_{\text{LOW}:\text{MEXT}}$

### Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for $t_{\text{IDLE}}$ greater than $t_{\text{HIGH},\text{MAX}}$. (refer to Table 366: I2C-SMBus specification data setup and hold times)

This timing parameter covers the condition where a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

#### 52.4.12 SMBus initialization

This section is relevant only when SMBus feature is supported. Refer to Section 52.3: I2C implementation.

In addition to I2C initialization, some other specific initialization must be done in order to perform SMBus communication:

**Received Command and Data Acknowledge control (Slave mode)**

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in slave mode, the Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. Refer to Slave Byte Control mode on page 1895 for more details.
Specific address (Slave mode)

The specific SMBus addresses must be enabled if needed. Refer to *Bus idle detection on page 1918* for more details.

- The SMBus Device Default Address (0b1100 001) is enabled by setting the SMBDEN bit in the I2C_CR1 register.
- The SMBus Host Address (0b0001 000) is enabled by setting the SMBHEN bit in the I2C_CR1 register.
- The Alert Response Address (0b0001100) is enabled by setting the ALERTEN bit in the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit in the I2C_CR1 register. Then the PEC transfer is managed with the help of a hardware byte counter: NBYTES[7:0] in the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in slave mode. The PEC is transferred after NBYTES-1 data have been transferred when the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

**Caution:** Changing the PECEN configuration is not allowed when the I2C is enabled.

<table>
<thead>
<tr>
<th>Table 373. SMBus with PEC configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode</strong></td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>Master Tx/Rx NBYTES + PEC + STOP</td>
</tr>
<tr>
<td>Master Tx/Rx NBYTES + PEC + ReSTART</td>
</tr>
<tr>
<td>Slave Tx/Rx with PEC</td>
</tr>
</tbody>
</table>

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits in the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

- **\( t_{\text{TIMEOUT}} \) check**
  
  In order to enable the \( t_{\text{TIMEOUT}} \) check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value in order to check the \( t_{\text{TIMEOUT}} \) parameter. The TIDLE bit must be configured to '0' in order to detect the SCL low level timeout.

  Then the timer is enabled by setting the TIMOUTEN in the I2C_TIMEOUTR register.

  If SCL is tied low for a time greater than (TIMEOUTA+1) x 2048 x \( t_{\text{I2CCLK}} \), the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 374: Examples of TIMEOUTA settings for various i2c_ker_ck frequencies (max \( t_{\text{TIMEOUT}} = 25 \text{ ms} \)).*  

**Caution:** Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMOUTEN bit is set.

- **\( t_{\text{LOW}:\text{SEXT}} \) and \( t_{\text{LOW}:\text{MEXT}} \) check**
  
  Depending on if the peripheral is configured as a master or as a slave, The 12-bit TIMEOUTB timer must be configured in order to check \( t_{\text{LOW}:\text{SEXT}} \) for a slave and
\(t_{\text{LOW;MEXT}}\) for a master. As the standard specifies only a maximum, the user can choose the same value for both.

Then the timer is enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for a time greater than \((\text{TIMEOUTB}+1) \times 2048 \times t_{\text{I2CCLK}}\), and in the timeout interval described in Bus idle detection on page 1918 section, the TIMEOUT flag is set in the I2C_ISR register.

Refer to Table 375: Examples of TIMEOUTB settings for various \(i2c\_ker\_ck\) frequencies

Caution: Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

**Bus Idle detection**

In order to enable the \(t_{\text{IDLE}}\) check, the 12-bit TIMEOUTA[11:0] field must be programmed with the timer reload value in order to obtain the \(t_{\text{IDLE}}\) parameter. The TIDLE bit must be configured to ‘1’ in order to detect both SCL and SDA high level timeout.

Then the timer is enabled by setting the TIMEOUTEN bit in the I2C_TIMEOUTR register.

If both the SCL and SDA lines remain high for a time greater than \((\text{TIMEOUTA}+1) \times 4 \times t_{\text{I2CCLK}}\), the TIMEOUT flag is set in the I2C_ISR register.

Refer to Table 376: Examples of TIMEOUTA settings for various \(i2c\_ker\_ck\) frequencies (max \(t_{\text{IDLE}} = 50\ \mu\text{s}\))

Caution: Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

**52.4.13 SMBus: I2C_TIMEOUTR register configuration examples**

This section is relevant only when SMBus feature is supported. Refer to Section 52.3: I2C implementation.

- Configuring the maximum duration of \(t_{\text{TIMEOUT}}\) to 25 ms:

<table>
<thead>
<tr>
<th>(f_{\text{I2CCLK}})</th>
<th>TIMEOUTA[11:0] bits</th>
<th>TIDLE bit</th>
<th>TIMEOUTEN bit</th>
<th>(t_{\text{TIMEOUT}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 MHz</td>
<td>(0x61)</td>
<td>0</td>
<td>1</td>
<td>(98 \times 2048 \times 125\ \text{ns} = 25\ \text{ms})</td>
</tr>
<tr>
<td>16 MHz</td>
<td>(0xC3)</td>
<td>0</td>
<td>1</td>
<td>(196 \times 2048 \times 62.5\ \text{ns} = 25\ \text{ms})</td>
</tr>
<tr>
<td>48 MHz</td>
<td>(0x249)</td>
<td>0</td>
<td>1</td>
<td>(586 \times 2048 \times 20.08\ \text{ns} = 25\ \text{ms})</td>
</tr>
</tbody>
</table>

- Configuring the maximum duration of \(t_{\text{LOW;SEXT}}\) and \(t_{\text{LOW;MEXT}}\) to 8 ms:

<table>
<thead>
<tr>
<th>(f_{\text{I2CCLK}})</th>
<th>TIMEOUTB[11:0] bits</th>
<th>TEXTEN bit</th>
<th>(t_{\text{LOW;EXT}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 MHz</td>
<td>(0x1F)</td>
<td>1</td>
<td>(32 \times 2048 \times 125\ \text{ns} = 8\ \text{ms})</td>
</tr>
<tr>
<td>16 MHz</td>
<td>(0x3F)</td>
<td>1</td>
<td>(64 \times 2048 \times 62.5\ \text{ns} = 8\ \text{ms})</td>
</tr>
<tr>
<td>48 MHz</td>
<td>(0xBB)</td>
<td>1</td>
<td>(188 \times 2048 \times 20.08\ \text{ns} = 8\ \text{ms})</td>
</tr>
</tbody>
</table>
• Configuring the maximum duration of \( t_{\text{IDLE}} \) to 50 µs

**Table 376. Examples of TIMEOUTA settings for various i2c_ker_ck frequencies**

<table>
<thead>
<tr>
<th>( f_{\text{I2CCLK}} )</th>
<th>TIMEOUTA[11:0] bits</th>
<th>TIDLE bit</th>
<th>TIMEOUTEN bit</th>
<th>( t_{\text{TIDLE}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 MHz</td>
<td>0x63</td>
<td>1</td>
<td>1</td>
<td>( 100 \times 4 \times 125 \text{ ns} = 50 \mu \text{s} )</td>
</tr>
<tr>
<td>16 MHz</td>
<td>0xC7</td>
<td>1</td>
<td>1</td>
<td>( 200 \times 4 \times 62.5 \text{ ns} = 50 \mu \text{s} )</td>
</tr>
<tr>
<td>48 MHz</td>
<td>0x257</td>
<td>1</td>
<td>1</td>
<td>( 600 \times 4 \times 20.08 \text{ ns} = 50 \mu \text{s} )</td>
</tr>
</tbody>
</table>

**52.4.14 SMBus slave mode**

This section is relevant only when SMBus feature is supported. Refer to Section 52.3: I2C implementation.

In addition to I2C slave transfer management (refer to Section 52.4.8: I2C slave mode) some additional software flowcharts are provided to support SMBus.

**SMBus Slave transmitter**

When the IP is used in SMBus, SBC must be programmed to ‘1’ in order to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts is NBYTES-1 and the content of the I2C_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES-1 data transfer.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.
Figure 529. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC

- Slave initialization
- I2C_ISR.ADDR = 1?
  - Read ADDCODE and DIR in I2C_ISR
  - I2C_CR2.NBYTES = N + 1
  - PECBYTE=1
  - Set I2C_ICR.ADDRCF
- I2C_ISR.TXIS = 1?
  - No
  - Write I2C_TXDR.TXDATA
  - SCL stretched
  - Yes

Figure 530. Transfer bus diagrams for SMBus slave transmitter (SBC=1)

Example SMBus slave transmitter 2 bytes + PEC,

- ADDR
- TXIS
- TXIS
- S Address A data1 A data2 P
- EV1 EV2 EV3
- NBYTES 3

EV1: ADDR ISR: check ADDCODE, program NBYTES=3, set PECBYTE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2

legend:
- transmission
- reception
- SCL stretch

MS19867V2

MS19869V2
SMBus Slave receiver

When the I2C is used in SMBus mode, SBC must be programmed to '1' in order to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD=1). Refer to Slave Byte Control mode on page 1895 for more details.

In order to check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTES-1 data have been received, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

If no ACK software control is needed, the user can program PECBYTE=1 and, in the same write operation, program NBYTES with the number of bytes to be received in a continuous flow. After NBYTES-1 are received, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.
Figure 531. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC

1. **Slave initialization**

   - **I2C_ISR.ADDR = 1?**
     - No
     - **I2C_ISR.RXNE = 1?**
       - No
       - **I2C_ISR.TCR = 1?**
         - Yes
         - Read ADDCODE and DIR in I2C_ISR
           - I2C_CR2.NBYTES = 1, RELOAD = 1
           - PECBYTE = 1
           - Set I2C_ICR.ADDRCF
           - Read I2C_RXDR.RXDATA
           - Program I2C_CR2.NACK = 0
           - I2C_CR2.NBYTES = 1
           - N = N - 1
           - **N = 1?**
             - No
             - **I2C_ISR.RXNE = 1?**
               - Yes
               - Read I2C_RXDR.RXDATA
               - Program RELOAD = 0
               - NACK = 0 and NBYTES = 1
               - **I2C_ISR.RXNE = 1?**
                 - Yes
                 - Read I2C_RXDR.RXDATA
                 - End
           - Yes
           - **I2C_ISR.RXNE = 1?**
             - Yes
             - Read I2C_RXDR.RXDATA
             - End
         - Yes
         - **I2C_ISR.ADDR = 1?**
           - Yes
           - **I2C_ISR.RXNE = 1?**
             - No
             - **I2C_ISR.TCR = 1?**
               - Yes
               - Read I2C_RXDR.RXDATA
               - Program I2C_CR2.NACK = 0
               - I2C_CR2.NBYTES = 1
               - N = N - 1
               - **N = 1?**
                 - No
                 - **I2C_ISR.RXNE = 1?**
                   - Yes
                   - Read I2C_RXDR.RXDATA
                   - End
                 - Yes
                 - **I2C_ISR.RXNE = 1?**
                   - Yes
                   - Read I2C_RXDR.RXDATA
                   - End
         - No
         - **I2C_ISR.RXNE = 1?**
           - Yes
           - Read I2C_RXDR.RXDATA
           - End
   - Yes
   - **SCL stretched**
Figure 532. Bus transfer diagrams for SMBus slave receiver (SBC=1)

Example SMBus slave receiver 2 bytes + PEC

Legend:
- transmission
- reception
- SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, program NBYTES = 3, PECBYTE=1, RELOAD=0, set ADDRCF
EV2: RXNE ISR: rd data1
EV3: RXNE ISR: rd data2
EV4: RXNE ISR: rd PEC

Example SMBus slave receiver 2 bytes + PEC, with ACK control (RELOAD=1/0)

Legend:
- transmission
- reception
- SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, program NBYTES = 1, PECBYTE=1, RELOAD=1, set ADDRCF
EV2: RXNE-TCR ISR: rd data1, program NACK=0 and NBYTES = 1
EV3: RXNE-TCR ISR: rd data2, program NACK=0, NBYTES = 1 and RELOAD=0
EV4: RXNE-TCR ISR: rd PEC

This section is relevant only when SMBus feature is supported. Refer to Section 52.3: I2C implementation.

In addition to I2C master transfer management (refer to Section 52.4.9: I2C master mode) some additional software flowcharts are provided to support SMBus.

**SMBus Master transmitter**

When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES-1. So if the PECBYTE bit is set when NBYTES=0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, automatic end mode must be selected (AUTOEND=1). In this case, the STOP condition automatically follows the PEC transmission.
When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND=0). In this case, once NBYTES-1 have been transmitted, the I2C_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 533. Bus transfer diagrams for SMBus master transmitter**

Example SMBus master transmitter 2 bytes + PEC, automatic end mode (STOP)

Example SMBus master transmitter 2 bytes + PEC, software end mode (RESTART)

INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECBYTE, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2

INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECBYTE, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START
SMBus Master receiver

When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND=1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND=0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.
Figure 534. Bus transfer diagrams for SMBus master receiver

Example SMBus master receiver 2 bytes + PEC, automatic end mode (STOP)

INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECBYTE, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2
EV3: RXNE ISR: rd PEC

Example SMBus master receiver 2 bytes + PEC, software end mode (RESTART)

INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECBYTE, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2
EV3: RXNE ISR: read PEC
EV4: TC ISR: program Slave address, program NBYTES = N, set START
52.4.15 Wakeup from Stop mode on address match

This section is relevant only when Wakeup from Stop mode feature is supported. Refer to Section 52.3: I2C implementation.

The I2C is able to wakeup the MCU from Stop mode (APB clock is off), when it is addressed. All addressing modes are supported.

Wakeup from Stop mode is enabled by setting the WUPEN bit in the I2C_CR1 register. The HSI or CSI oscillator must be selected as the clock source for I2CCLK in order to allow wakeup from Stop mode.

During Stop mode, the HSI or CSI is switched off. When a START is detected, the I2C interface switches the HSI or CSI on, and stretches SCL low until HSI or CSI is woken up.

HSI or CSI is then used for the address reception.

In case of an address match, the I2C stretches SCL low during MCU wakeup time. The stretch is released when ADDR flag is cleared by software, and the transfer goes on normally.

If the address does not match, the HSI or CSI is switched off again and the MCU is not woken up.

Note: If the I2C clock is the system clock, or if WUPEN = 0, the HSI or CSI is not switched on after a START is received.

Only an ADDR interrupt can wakeup the MCU. Therefore do not enter Stop mode when the I2C is performing a transfer as a master, or as an addressed slave after the ADDR flag is set. This can be managed by clearing SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

Caution: The digital filter is not compatible with the wakeup from Stop mode feature. If the DNF bit is not equal to 0, setting the WUPEN bit has no effect.

Caution: This feature is available only when the I2C clock source is the HSI or CSI oscillator.

Caution: Clock stretching must be enabled (NOSTRETCH=0) to ensure proper operation of the wakeup from Stop mode feature.

Caution: If wakeup from Stop mode is disabled (WUPEN=0), the I2C peripheral must be disabled before entering Stop mode (PE=0).

52.4.16 Error conditions

The following errors are the error conditions which may cause communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of 9 SCL clock pulses. A START or a STOP condition is detected when a SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (i.e not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.
Arbitration lost (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In master mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the master switches automatically to slave mode.
- In slave mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in slave mode when NOSTRETCH=1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF=1 and the first data byte should be sent. The content of the I2C_TXDR register is sent if TXE=0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Packet Error Checking Error (PECERR)

This section is relevant only when the SMBus feature is supported. Refer to Section 52.3: I2C implementation.

A PEC error is detected when the received PEC byte does not match with the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Timeout Error (TIMEOUT)

This section is relevant only when the SMBus feature is supported. Refer to Section 52.3: I2C implementation.

A timeout error occurs for any of these conditions:

- TIDLE=0 and SCL remained low for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a SMBus timeout.
- TIDLE=1 and both SDA and SCL remained high for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus $t_{LOW:MEXT}$ parameter)
- Slave cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus $t_{LOW:SEXT}$ parameter)
When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

**Alert (ALERT)**

This section is relevant only when the SMBus feature is supported. Refer to [Section 52.3: I2C implementation](#).

The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN=1), the alert pin detection is enabled (ALERTEN=1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

### 52.4.17 DMA requests

#### Transmission using DMA

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the I2C_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see [Section 15: Direct memory access controller (DMA) on page 592](#)) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to [Master transmitter on page 1906](#).
- In slave mode:
  - With NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOSTRETCH=1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to [SMBus Slave transmitter on page 1921](#) and [SMBus Master transmitter on page 1925](#).

**Note:** If DMA is used for transmission, the TXIE bit does not need to be enabled.

#### Reception using DMA

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured using the DMA peripheral (refer to [Section 15: Direct memory access controller (DMA) on page 592](#)) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the
DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

- In slave mode with NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see Section 52.3: I2C implementation): the PEC transfer is managed with the NBYTES counter. Refer to SMBus Slave receiver on page 1923 and SMBus Master receiver on page 1927.

Note: If DMA is used for reception, the RXIE bit does not need to be enabled.

52.4.18 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG_I2Cx_ configuration bits in the DBG module.

52.5 I2C low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. I2C interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop(1)</td>
<td>The I2C registers content is kept. If WUPEN = 1 and I2C is clocked by an internal oscillator (HSI or CSI): the address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. If WUPEN=0: the I2C must be disabled before entering Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The I2C peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

1. Refer to I2C implementation table for information about the Stop modes supported by each instance. If wakeup from a specific Stop mode is not supported, the instance must be disabled before entering this Stop mode.
### 52.6 I2C interrupts

The table below gives the list of I2C interrupt requests.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Event flag/Interrupt clearing method</th>
<th>Interrupt enable control bit</th>
<th>Interrupt/Wakeup activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive buffer not empty</td>
<td>RXNE</td>
<td>Read I2C_RXDR register</td>
<td>RXIE</td>
<td>i2c_event_it</td>
</tr>
<tr>
<td>Transmitter buffer interrupt status</td>
<td>TXIS</td>
<td>Write I2C_TXDR register</td>
<td>TXIS</td>
<td>i2c_error_it</td>
</tr>
<tr>
<td>Stop detection interrupt flag</td>
<td>STOPF</td>
<td>Write STOPCF=1</td>
<td>STOPIE</td>
<td>i2c_wkup</td>
</tr>
<tr>
<td>Transfer Complete Reload</td>
<td>TCR</td>
<td>Write I2C_CR2 with NBYTES[7:0] ≠ 0</td>
<td>TCIE</td>
<td></td>
</tr>
<tr>
<td>Transfer complete</td>
<td>TC</td>
<td>Write START=1 or STOP=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address matched</td>
<td>ADDR</td>
<td>Write ADDRCF=1</td>
<td>ADDRIE</td>
<td></td>
</tr>
<tr>
<td>NACK reception</td>
<td>NACKF</td>
<td>Write NACKCF=1</td>
<td>NACKIE</td>
<td></td>
</tr>
<tr>
<td>Bus error</td>
<td>BERR</td>
<td>Write BERRCF=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arbitration loss</td>
<td>ARLO</td>
<td>Write ARLOCF=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overrun/Underrun</td>
<td>OVR</td>
<td>Write OVRCF=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEC error</td>
<td>PECERR</td>
<td>Write PECERRCF=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timeout/tLOW error</td>
<td>TIMEOUT</td>
<td>Write TIMEOUTCF=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMBus Alert</td>
<td>ALERT</td>
<td>Write ALERTCF=1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. If WUPEN is set.
52.7 I2C registers

Refer to Section 1.2 on page 97 for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

52.7.1 I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

```
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-24</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>PECEN: PEC enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>0:</td>
<td>PEC calculation disabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1:</td>
<td>PEC calculation enabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 52.3: I2C implementation.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Bit 22  **ALERTEN**: SMBus alert enable

0: The SMBus alert pin (SMBA) is not supported in host mode (SMBHEN=1). In device mode (SMBHEN=0), the SMBA pin is released and the Alert Response Address header is disabled (0001100x followed by NACK).

1: The SMBus alert pin is supported in host mode (SMBHEN=1). In device mode (SMBHEN=0), the SMBA pin is driven low and the Alert Response Address header is enabled (0001100x followed by ACK).

*Note:*  When ALERTEN=0, the SMBA pin can be used as a standard GPIO. If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 52.3: I2C implementation.

Bit 21  **SMBDEN**: SMBus Device Default Address enable

0: Device Default Address disabled. Address 0b1100001x is NACKed.

1: Device Default Address enabled. Address 0b1100001x is ACKed.

*Note:* If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 52.3: I2C implementation.

Bit 20  **SMBHEN**: SMBus Host Address enable

0: Host Address disabled. Address 0b0001000x is NACKed.

1: Host Address enabled. Address 0b0001000x is ACKed.

*Note:* If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 52.3: I2C implementation.

Bit 19  **GCEN**: General call enable

0: General call disabled. Address 0b00000000 is NACKed.

1: General call enabled. Address 0b00000000 is ACKed.

Bit 18  **WUPEN**: Wakeup from Stop mode enable

0: Wakeup from Stop mode disable.

1: Wakeup from Stop mode enable.

*Note:* If the Wakeup from Stop mode feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 52.3: I2C implementation.

*Note:* WUPEN can be set only when DNF = '0000'

Bit 17  **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.

0: Clock stretching enabled

1: Clock stretching disabled

*Note:* This bit can only be programmed when the I2C is disabled (PE = 0).

Bit 16  **SBC**: Slave byte control

This bit is used to enable hardware byte control in slave mode.

0: Slave byte control disabled

1: Slave byte control enabled

Bit 15  **RXDMAEN**: DMA reception requests enable

0: DMA mode disabled for reception

1: DMA mode enabled for reception

Bit 14  **TXDMAEN**: DMA transmission requests enable

0: DMA mode disabled for transmission

1: DMA mode enabled for transmission

Bit 13  Reserved, must be kept at reset value.
Bit 12 **ANFOFF**: Analog noise filter OFF
- 0: Analog noise filter enabled
- 1: Analog noise filter disabled

*Note:* This bit can only be programmed when the I2C is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] \* tI2CCLK
- 0000: Digital filter disabled
- 0001: Digital filter enabled and filtering capability up to 1 tI2CCLK
  - ... 1111: digital filter enabled and filtering capability up to 15 tI2CCLK

*Note:* If the analog filter is also enabled, the digital filter is added to the analog filter. This filter can only be programmed when the I2C is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable
- 0: Error detection interrupts disabled
- 1: Error detection interrupts enabled

*Note:* Any of these errors generate an interrupt:
- Arbitration Loss (ARLO)
- Bus Error detection (BERR)
- Overrun/Underrun (OVR)
- Timeout detection (TIMEOUT)
- PEC error detection (PECERR)
- Alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer Complete interrupt enable
- 0: Transfer Complete interrupt disabled
- 1: Transfer Complete interrupt enabled

*Note:* Any of these events generate an interrupt:
- Transfer Complete (TC)
- Transfer Complete Reload (TCR)

Bit 5 **STOPIE**: Stop detection Interrupt enable
- 0: Stop detection (STOPF) interrupt disabled
- 1: Stop detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received Interrupt enable
- 0: Not acknowledge (NACKF) received interrupts disabled
- 1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match Interrupt enable (slave only)
- 0: Address match (ADDR) interrupts disabled
- 1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX Interrupt enable
- 0: Receive (RXNE) interrupts disabled
- 1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX Interrupt enable
- 0: Transmit (TXIS) interrupt disabled
- 1: Transmit (TXIS) interrupt enabled
Bit 0 **PE**: Peripheral enable  
0: Peripheral disable  
1: Peripheral enable  

*Note: When PE=0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.*

### 52.7.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

#### Bit 0 PE: Peripheral enable

0: Peripheral disable  
1: Peripheral enable  

*Note: When PE=0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.*

<table>
<thead>
<tr>
<th>Bit 31:27</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

#### Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE=0.  
0: No PEC transfer.  
1: PEC transmission/reception is requested

*Note: Writing ‘0’ to this bit has no effect.*

This bit has no effect when RELOAD is set.  
This bit has no effect is slave mode when SBC=0.  
*If the SMBus feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 52.3: I2C implementation.*

#### Bit 25 **AUTOEND**: Automatic end mode (master mode)

This bit is set and cleared by software.  
0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.  
1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

*Note: This bit has no effect in slave mode or when the RELOAD bit is set.*

#### Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.  
0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).  
1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.
Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with SBC=0.

*Note:* Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (slave mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE=0.

0: an ACK is sent after current received byte.
1: a NACK is sent after current received byte.

*Note:* Writing '0' to this bit has no effect.

This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE=1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: Stop generation (master mode)

The bit is set by software, cleared by hardware when a STOP condition is detected, or when PE = 0.

In Master Mode:

0: No Stop generation.
1: Stop generation after current byte transfer.

*Note:* Writing '0' to this bit has no effect.

Bit 13 **START**: Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software by writing ‘1’ to the ADDRCF bit in the I2C_ICR register.

0: No Start generation.
1: Restart/Start generation:

If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated Start condition when RELOAD=0, after the end of the NBYTES transfer. Otherwise setting this bit generates a START condition once the bus is free.

*Note:* Writing '0' to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in slave mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (master receiver mode)

0: The master sends the complete 10 bit slave address read sequence: Start + 2 bytes 10bit address in write direction + Restart + 1st 7 bits of the 10 bit address in read direction.
1: The master only sends the 1st 7 bits of the 10 bit address, followed by Read direction.

*Note:* Changing this bit when the START bit is set is not allowed.
Bit 11 **ADD10**: 10-bit addressing mode (master mode)
  0: The master operates in 7-bit addressing mode,
  1: The master operates in 10-bit addressing mode
  
  *Note:* Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (master mode)
  0: Master requests a write transfer.
  1: Master requests a read transfer.
  
  *Note:* Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Slave address (master mode)

  **In 7-bit addressing mode (ADD10 = 0):**
  SADD[7:1] should be written with the 7-bit slave address to be sent. The bits SADD[9],
  SADD[8] and SADD[0] are don't care.

  **In 10-bit addressing mode (ADD10 = 1):**
  SADD[9:0] should be written with the 10-bit slave address to be sent.

  *Note:* Changing these bits when the START bit is set is not allowed.
52.7.3  I2C own address 1 register (I2C_OAR1)

Address offset: 0x08
Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bit 15  **OA1EN**: Own Address 1 enable
0: Own address 1 disabled. The received slave address OA1 is NACKed.
1: Own address 1 enabled. The received slave address OA1 is ACKed.

Bits 14:11  Reserved, must be kept at reset value.

Bit 10  **OA1MODE**: Own Address 1 10-bit mode
0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.

*Note: This bit can be written only when OA1EN=0.*

Bits 9:0  **OA1[9:0]**: Interface own slave address
7-bit addressing mode: OA1[7:1] contains the 7-bit own slave address. The bits OA1[9], OA1[8] and OA1[0] are don't care.
10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address.

*Note: These bits can be written only when OA1EN=0.*
52.7.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-16</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td>0x0000 0000</td>
</tr>
<tr>
<td>15</td>
<td>OA2EN: Own Address 2 enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Own address 2 disabled. The received slave address OA2 is NACKed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Own address 2 enabled. The received slave address OA2 is ACKed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14-11</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>10-8</td>
<td>OA2MSK[2:0]: Own Address 2 masks</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000: No mask</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>001: OA2[1] is masked and don’t care. Only OA2[7:2] are compared.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>010: OA2[2:1] are masked and don’t care. Only OA2[7:3] are compared.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>100: OA2[4:1] are masked and don’t care. Only OA2[7:5] are compared.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>111: OA2[7:1] are masked and don’t care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: These bits can be written only when OA2EN=0.

As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxxx and 0b1111xxx) are not acknowledged even if the comparison matches.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-1</td>
<td>OA2[7:1]: Interface address</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7-bit addressing mode: 7-bit address</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: These bits can be written only when OA2EN=0.

Bit 0 Reserved, must be kept at reset value.
52.7.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10
Reset value: 0x0000 0000
Access: No wait states

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10</td>
<td>rw</td>
<td>Timing prescaler</td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>Data setup time</td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>Data hold time</td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>SCL high period (master mode)</td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>SCL low period (master mode)</td>
</tr>
</tbody>
</table>

**Bits 31:28 PRESC[3:0]:** Timing prescaler

This field is used to prescale i2c_ker_ck in order to generate the clock period \( t_{\text{PRESC}} \) used for data setup and hold counters (refer to I2C timings on page 1887) and for SCL high and low level counters (refer to I2C master initialization on page 1902).

\[ t_{\text{PRESC}} = (\text{PRESC}+1) \times t_{\text{I2CCLK}} \]

**Bits 27:24 Reserved, must be kept at reset value.**

**Bits 23:20 SCLDEL[3:0]:** Data setup time

This field is used to generate a delay \( t_{\text{SCLDEL}} \) between SDA edge and SCL rising edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during \( t_{\text{SCLDEL}} \).

\[ t_{\text{SCLDEL}} = (\text{SCLDEL}+1) \times t_{\text{PRESC}} \]

**Note:** \( t_{\text{SCLDEL}} \) is used to generate \( t_{\text{SU}:\text{DAT}} \) timing.

**Bits 19:16 SDADEL[3:0]:** Data hold time

This field is used to generate the delay \( t_{\text{SDADEL}} \) between SCL falling edge and SDA edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during \( t_{\text{SDADEL}} \).

\[ t_{\text{SDADEL}} = \text{SDADEL} \times t_{\text{PRESC}} \]

**Note:** \( \text{SDADEL} \) is used to generate \( t_{\text{HD}:\text{DAT}} \) timing.

**Bits 15:8 SCLH[7:0]:** SCL high period (master mode)

This field is used to generate the SCL high period in master mode.

\[ t_{\text{SCLH}} = (\text{SCLH}+1) \times t_{\text{PRESC}} \]

**Note:** \( \text{SCLH} \) is also used to generate \( t_{\text{SU}:\text{STO}} \) and \( t_{\text{HD}:\text{STA}} \) timing.

**Bits 7:0 SCLL[7:0]:** SCL low period (master mode)

This field is used to generate the SCL low period in master mode.

\[ t_{\text{SCLL}} = (\text{SCLL}+1) \times t_{\text{PRESC}} \]

**Note:** \( \text{SCLL} \) is also used to generate \( t_{\text{BUF}} \) and \( t_{\text{SU}:\text{STA}} \) timings.

**Note:** This register must be configured when the I2C is disabled (PE = 0).

**Note:** The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.
### 52.7.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
</table>
| 31  | TEXTEN: Extended clock timeout enable | 0: Extended clock timeout detection is disabled  
1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT=1). |
| 30-28 | Reserved                  | 000            |
| 27-26 | TIMEOUTB[11:0]              | 00             |
| 25  | TIDLE: Idle clock timeout detection | 0: TIMEOUTA is used to detect SCL low timeout  
1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition) |
| 24-13 | Reserved                  | 0000           |
| 12   | TIMOUTEN: Clock timeout enable | 0: SCL timeout detection is disabled  
1: SCL timeout detection is enabled: when SCL is low for more than $t_{TIMEOUT}$ (TIDLE=0) or high for more than $t_{IDLE}$ (TIDLE=1), a timeout error is detected (TIMEOUT=1). |
| 11-0 | TIMEOUTA[11:0]: Bus Timeout A | 00000          |

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to “0x00000000”. Refer to Section 52.3: I2C implementation.
### 52.7.7 I2C interrupt and status register (I2C_ISR)

- **Address offset:** 0x18
- **Reset value:** 0x0000 0001
- **Access:** No wait states

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>BUSY</td>
<td>Res.</td>
<td>ALERT</td>
<td>TIME</td>
<td>OUT</td>
<td>PEC</td>
<td>ERR</td>
<td>OVR</td>
<td>ARLO</td>
<td>BERR</td>
<td>TCR</td>
<td>TC</td>
<td>STOPF</td>
<td>NACKF</td>
<td>ADDR</td>
<td>RXNE</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>rs</td>
<td>rs</td>
<td></td>
</tr>
</tbody>
</table>

- **Bits 31:24** Reserved, must be kept at reset value.
- **Bits 23:17** **ADDRCODE[6:0]:** Address match code (Slave mode)
  These bits are updated with the received address when an address match event occurs (ADDR = 1).
  In the case of a 10-bit address, ADDRCODE provides the 10-bit header followed by the 2 MSBs of the address.
- **Bit 16** **DIR:** Transfer direction (Slave mode)
  This flag is updated when an address match event occurs (ADDR=1).
  0: Write transfer, slave enters receiver mode.
  1: Read transfer, slave enters transmitter mode.
- **Bit 15** **BUSY:** Bus busy
  This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected, or when PE=0.
- **Bit 14** Reserved, must be kept at reset value.
- **Bit 13** **ALERT:** SMBus alert
  This flag is set by hardware when SMBHEN=1 (SMBus host configuration), ALERTEN=1 and a SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.
  **Note:** This bit is cleared by hardware when PE=0.
  If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 52.3: I2C implementation.
- **Bit 12** **TIMEOUT:** Timeout or t\textsubscript{LOW} detection flag
  This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.
  **Note:** This bit is cleared by hardware when PE=0.
  If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 52.3: I2C implementation.
Bit 11 **PECERR**: PEC Error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

*Note:* This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 52.3: I2C implementation.

Bit 10 **OVR**: Overrun/Underrun (slave mode)

This flag is set by hardware in slave mode with NOSTRETCH=1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

*Note:* This bit is cleared by hardware when PE=0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

*Note:* This bit is cleared by hardware when PE=0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced Start or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting BERRCF bit.

*Note:* This bit is cleared by hardware when PE=0.

Bit 7 **TCR**: Transfer Complete Reload

This flag is set by hardware when RELOAD=1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

*Note:* This bit is cleared by hardware when PE=0.

This flag is only for master mode, or for slave mode when the SBC bit is set.

Bit 6 **TC**: Transfer Complete (master mode)

This flag is set by hardware when RELOAD=0, AUTOEND=0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

*Note:* This bit is cleared by hardware when PE=0.

Bit 5 **STOPF**: Stop detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

– either as a master, provided that the STOP condition is generated by the peripheral.

– or as a slave, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

*Note:* This bit is cleared by hardware when PE=0.

Bit 4 **NACKF**: Not Acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

*Note:* This bit is cleared by hardware when PE=0.

Bit 3 **ADDR**: Address matched (slave mode)

This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting ADDRCF bit.

*Note:* This bit is cleared by hardware when PE=0.
52.7.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C
Reset value: 0x0000 0000
Access: No wait states

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

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 ALERTCF: Alert flag clear
Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 52.3: I2C implementation.

Bit 12 TIMOUTCF: Timeout detection flag clear
Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 52.3: I2C implementation.

Bit 11 PECCF: PEC Error flag clear
Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 52.3: I2C implementation.
Bit 10 **OVRCF**: Overrun/Underrun flag clear  
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear  
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear  
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear  
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not Acknowledge flag clear  
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear  
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

### 52.7.9 I2C PEC register (I2C_PECR)

Address offset: 0x20  
Reset value: 0x0000 0000  
Access: No wait states

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

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

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

Bits 7:0 **PEC[7:0]** Packet error checking register  
This field contains the internal PEC when PECEN=1.  
The PEC is cleared by hardware when PE=0.

**Note:** If the SMBus feature is not supported, this register is reserved and forced by hardware to “0x00000000”. Refer to Section 52.3: I2C implementation.
52.7.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24
Reset value: 0x0000 0000
Access: No wait states

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

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

Bits 7:0 **RXDATA[7:0]** 8-bit receive data
Data byte received from the I2C bus

52.7.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28
Reset value: 0x0000 0000
Access: No wait states

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

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

Bits 7:0 **TXDATA[7:0]** 8-bit transmit data
Data byte to be transmitted to the I2C bus

*Note: These bits can be written only when TXE=1.*
### 52.7.12 I2C register map

The table below provides the I2C register map and reset values.

| Offset | Register name | Offset 31 | Offset 30 | Offset 29 | Offset 28 | Offset 27 | Offset 26 | Offset 25 | Offset 24 | Offset 23 | Offset 22 | Offset 21 | Offset 20 | Offset 19 | Offset 18 | Offset 17 | Offset 16 | Offset 15 | Offset 14 | Offset 13 | Offset 12 | Offset 11 | Offset 10 | Offset 9 | Offset 8 | Offset 7 | Offset 6 | Offset 5 | Offset 4 | Offset 3 | Offset 2 | Offset 1 | Offset 0 |
|--------|---------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 0x0    | I2C_CR1       | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         |
| Reset value | | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x4    | I2C_CR2       | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         |
| Reset value | | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x8    | I2C_OAR1      | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         |
| Reset value | | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0xC    | I2C_OAR2      | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         |
| Reset value | | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x10   | I2C_TIMINGR   | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         |
| Reset value | | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x14   | I2C_TIMEOUTR  | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         |
| Reset value | | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x18   | I2C_ISR       | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         |
| Reset value | | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 |
| 0x1C   | I2C_ICR       | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         |
| Reset value | | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x20   | I2C_PECR      | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         |
| Reset value | | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x24   | I2C_RXDR      | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         |
| Reset value | | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
Refer to Section 2.3 on page 128 for the register boundary addresses.

Table 379. I2C register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x28   | I2C_TXDR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 on page 128 for the register boundary addresses.
53 Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

53.1 USART introduction

The USART offers a flexible means to perform Full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and Half-duplex Single-wire communications, as well as LIN (local interconnection network), Smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and Modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.
53.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data
  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wakeup from Mute mode by idle line detection or address mark detection
53.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T=0 and T=1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for Smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

53.4 USART implementation

Table 380 describes USART implementation on STM32H7A3/7B3 and STM32H7B0 devices.

<table>
<thead>
<tr>
<th>USART modes/features(1)</th>
<th>USART1/2/3/6/10</th>
<th>UART4/5/7/8/9</th>
<th>LPUART1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware flow control for modem</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Continuous communication using DMA</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Multiprocessor communication</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Synchronous mode (Master/Slave)</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Smartcard mode</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Single-wire Half-duplex communication</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IrDA SIR ENDEC block</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>LIN mode</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Dual clock domain and wakeup from low-power mode</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Receiver timeout interrupt</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Modbus communication</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Auto baud rate detection</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Driver Enable</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>USART data length</td>
<td>7, 8 and 9 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tx/Rx FIFO</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Tx/Rx FIFO size</td>
<td>16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. X = supported.
53.5 **USART functional description**

53.5.1 **USART block diagram**

The simplified block diagram given in Figure 536 shows two fully-independent clock domains:

- **The usart_pclk clock domain**
  
The usart_pclk clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.

- **The usart_ker_ck kernel clock domain.**
  
The usart_ker_ck is the USART clock source. It is independent from usart_pclk and delivered by the RCC. The USART registers can consequently be written/read even when the usart_ker_ck clock is stopped.
  
  When the dual clock domain feature is disabled, the usart_ker_ck clock is the same as the usart_pclk clock.

There is no constraint between usart_pclk and usart_ker_ck: usart_ker_ck can be faster or slower than usart_pclk. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external SCLK signal provided by the external master SPI device. The usart_ker_ck clock must be at least 3 times faster than the clock on the CK input.
53.5.2 USART signals

USART bidirectional communications

USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)
  RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

- **TX** (Transmit Data Output)
  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In Single-wire and Smartcard modes, this I/O is used to transmit and receive data.

RS232 Hardware flow control mode

The following pins are required in RS232 Hardware flow control mode:

- **CTS** (Clear To Send)
  When driven high, this signal blocks the data transmission at the end of the current transfer.

- **RTS** (Request To Send)
  When it is low, this signal indicates that the USART is ready to receive data.

RS485 Hardware control mode

The following pin is required in RS485 Hardware control mode:

- **DE** (Driver Enable)
  This signal activates the transmission mode of the external transceiver.

*Note:* *DE and RTS share the same pin.*

Synchronous master/slave mode and Smartcard mode

The following pin is required in synchronous master/slave mode and Smartcard mode:

- **CK**
  This pin acts as Clock output in Synchronous master and Smartcard modes. It acts as Clock input is Synchronous slave mode.
  In Synchronous Master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable.
  In Smartcard mode, CK output provides the clock to the smartcard.

- **NSS**
  This pin acts as Slave Select input in Synchronous slave mode.

*Note:* *NSS and CTS share the same pin.*
53.5.3 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see Figure 537):

- 7-bit character length: M[1:0] = ’10’
- 8-bit character length: M[1:0] = ’00’
- 9-bit character length: M[1:0] = ’01’

Note: In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

An **Idle character** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A **Break character** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.
Figure 537. Word length programming

9-bit word length (M = 01), 1 Stop bit

Data frame

Possible
Parity
bit

Start
bit

Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Bit8

Stop
bit

Start
bit

**

Idle frame

Break frame

8-bit word length (M = 00), 1 Stop bit

Data frame

Possible
Parity
bit

Start
bit

Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7

Stop
bit

Start
bit

**

Idle frame

Break frame

7-bit word length (M = 10), 1 Stop bit

Data frame

Possible
Parity
bit

Start
bit

Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6

Stop
bit

Start
bit

**

Idle frame

Break frame

** LBCL bit controls last data clock pulse

MS33194V2
53.5.4 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and Smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in USART_CR3 control register.

In this case:

- The RXFT flag is set in the USART_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.
  
  This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.

  RXFTCFG data have been received: one data in USART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to ’101’, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size -1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data is not set the overrun flag.

- The TXFT flag is set in the USART_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.
  
  This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

53.5.5 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the SCLK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.
Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

**Note:**
The TE bit must be set before writing the data to be transmitted to the USART_TDR.
The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

**Configurable stop bits**
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13, 12.

- **1 stop bit:** This is the default value of number of stop bits.
- **2 stop bits:** This is supported by normal USART, Single-wire and Modem modes.
- **1.5 stop bits:** To be used in Smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission features 10 low bits (when M[1:0] = ‘00’) or 11 low bits (when M[1:0] = ‘01’) or 9 low bits (when M[1:0] = ‘10’) followed by 2 stop bits (see Figure 538). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

**Figure 538. Configurable stop bits**

[Diagram showing configurations for 8-bit data with 1, 1/2, and 2 stop bits]
Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in Section 53.5.10: USART multiprocessor communication.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   - When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   - When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC=1.
   - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
   - When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.
Single byte communication

- When FIFO mode is disabled
  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:
  - the data have been moved from the USART_TDR register to the shift register and the data transmission has started;
  - the USART_TDR register is empty;
  - the next data can be written to the USART_TDR register without overwriting the previous data.
  This flag generates an interrupt if the TXEIE bit is set.
  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.
  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  - the TXFIFO is not full;
  - the USART_TDR register is empty;
  - the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.
  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.
  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.
  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the microcontroller to enter the low-power mode (see Figure 539: TC/TXE behavior when transmitting).
Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see Figure 537).

If a ‘1’ is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

53.5.6 USART receiver

The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0 X 0 X 0 X 0 X.
Figure 540. Start bit detection when oversampling by 16 or 8

<table>
<thead>
<tr>
<th>RX state</th>
<th>Idle</th>
<th>Start bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX line</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ideal sample clock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real sample clock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conditions to validate the start bit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE=1, or RXFNE flag set and interrupt generated if RXFNEIE=1 if FIFO mode enabled) if the 3 sampled bits are at '0' (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at '0' and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at '0').

The start bit is validated but the NE noise flag is set if,

a) for both samplings, 2 out of the 3 sampled bits are at '0' (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)

or

b) for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at '0'.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).
Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to ‘1’.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in Section 53.5.10: USART multiprocessor communication.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty i.e. when there are data to be read from the RXFIFO.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to ‘1’ in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to ‘1’ in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be
generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

**Break character**

When a break character is received, the USART handles it as a framing error.

**Idle character**

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

**Overrun error**

- **FIFO mode disabled**
  
  An overrun error occurs if a character is received and RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte reception. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
  
  - the ORE bit is set;
  - the RDR content is not lost. The previous data is available by reading the USART_RDR register.
  - the shift register is overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXNEIE or the EIE bit is set.

- **FIFO mode enabled**
  
  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full. Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty. An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:
  
  - The ORE bit is set.
  - The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the OREC bit in the USART_ICR register.

*Note:* The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- if RXNE=1, then the last valid data is stored in the receive register (RDR) and can be read,
- if RXNE=0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.
Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see Section Reset and clock control (RCC)). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is `usart_ker_ck`.

When the dual clock domain and the wakeup from low-power mode features are supported, the `usart_ker_ck` clock source can be configurable in the RCC (see Section Reset and clock control (RCC)). Otherwise the `usart_ker_ck` clock is the same as `usart_pclk`.

The `usart_ker_ck` clock can be divided by a programmable factor, defined in the USART_PRES register.

Some `usart_ker_ck` sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see Figure 542 and Figure 543).

Depending on your application:

- select oversampling by 8 (OVER8=1) to achieve higher speed (up to `usart_ker_ck_pres/8`). In this case the maximum receiver tolerance to clock deviation is reduced (refer to Section 53.5.8: Tolerance of the USART receiver to clock deviation on page 1970)
- select oversampling by 16 (OVER8=0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum...
Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

Depending on your application:

- select the three sample majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to Figure 381) because this indicates that a glitch occurred during the sampling.
- select the single sample method (ONEBIT=1) when the line is noise-free to increase the receiver tolerance to clock deviations (see Section 53.5.8: Tolerance of the USART receiver to clock deviation on page 1970). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NECF bit in USART_ICR register.

**Note:** Noise error is not supported in SPI mode.

Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to ‘0’ by hardware.

**Figure 542. Data sampling when oversampling by 16**
Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:
- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing ‘1’ to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.
Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.

- **0.5 stop bit (reception in Smartcard mode):** no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.

- **1 stop bit:** sampling for 1 stop bit is done on the 8th, 9th and 10th samples.

- **1.5 stop bits (Smartcard mode):**
  When transmitting in Smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE =1 in USART_CR1) and the stop bit is checked to test if the Smartcard has detected a parity error.

In the event of a parity error, the Smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to Section 53.5.16: USART receiver timeout on page 1984 for more details).

- **2 stop bits:**
  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. The framing error flag is set if a framing error is detected during the first stop bit. The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

### 53.5.7 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

**Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = ‘0’ or ‘1’)***

In case of oversampling by 16, the baud rate is given by the following formula:

\[
Tx/Rx \text{ baud} = \frac{\text{usart_ker_ckpres}}{\text{USARTDIV}}
\]

In case of oversampling by 8, the baud rate is given by the following formula:

\[
Tx/Rx \text{ baud} = 2 \times \frac{\text{usart_ker_ckpres}}{\text{USARTDIV}}
\]

**Equation 2: baud rate in Smartcard, LIN and IrDA modes (OVER8 = 0)***

The baud rate is given by the following formula:

\[
Tx/Rx \text{ baud} = \frac{\text{usart_ker_ckpres}}{\text{USARTDIV}}
\]
USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value should not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1
To obtain 9600 baud with usart_ker_ck_pres = 8 MHz:

- In case of oversampling by 16:
  \[ \text{USARTDIV} = \frac{8\,000\,000}{9600} \]
  \[ \text{BRR} = \text{USARTDIV} = 833d = 0341h \]

- In case of oversampling by 8:
  \[ \text{USARTDIV} = \frac{2 \times 8\,000\,000}{9600} \]
  \[ \text{USARTDIV} = 1666,66 (1667d = 683h) \]
  \[ \text{BRR}[3:0] = 3h >> 1 = 1h \]
  \[ \text{BRR} = 0x681 \]

Example 2
To obtain 921.6 Kbaud with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  \[ \text{USARTDIV} = \frac{48\,000\,000}{921\,600} \]
  \[ \text{BRR} = \text{USARTDIV} = 52d = 34h \]

- In case of oversampling by 8:
  \[ \text{USARTDIV} = \frac{2 \times 48\,000\,000}{921\,600} \]
  \[ \text{USARTDIV} = 104 (104d = 68h) \]
  \[ \text{BRR}[3:0] = \text{USARTDIV}[3:0] >> 1 = 8h >> 1 = 4h \]
  \[ \text{BRR} = 0x64 \]

53.5.8 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.
The causes which contribute to the total deviation are:

- **DTRA**: deviation due to the transmitter error (which also includes the deviation of the transmitter’s local oscillator)
- **DQUANT**: error due to the baud rate quantization of the receiver
- **DREC**: deviation of the receiver local oscillator
- **DTCL**: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

\[
\text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} + \text{DWU} < \text{USART receiver tolerance}
\]

where

\[\text{DWU} = \begin{cases} 
\frac{t_{\text{WUUSART}}}{11 \times T\text{bit}} & \text{when } M[1:0] = 01; \\
\frac{t_{\text{WUUSART}}}{10 \times T\text{bit}} & \text{when } M[1:0] = 00; \\
\frac{t_{\text{WUUSART}}}{9 \times T\text{bit}} & \text{when } M[1:0] = 10;
\end{cases}\]

\(t_{\text{WUUSART}}\) is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in Table 382, Table 383, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_C1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_C1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_C3 register.

**Table 382. Tolerance of the USART receiver when BRR [3:0] = 0000**

<table>
<thead>
<tr>
<th>M bits</th>
<th>OVER8 bit = 0</th>
<th>OVER8 bit = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ONEBIT=0</td>
<td>ONEBIT=1</td>
</tr>
<tr>
<td>00</td>
<td>3.75%</td>
<td>4.375%</td>
</tr>
<tr>
<td>01</td>
<td>3.41%</td>
<td>3.97%</td>
</tr>
<tr>
<td>10</td>
<td>4.16%</td>
<td>4.86%</td>
</tr>
</tbody>
</table>
53.5.9 USART Auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

### Table 383. Tolerance of the USART receiver when BRR[3:0] is different from 0000

<table>
<thead>
<tr>
<th>M bits</th>
<th>OVER8 bit = 0</th>
<th>OVER8 bit = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ONEBIT=0</td>
<td>ONEBIT=1</td>
</tr>
<tr>
<td>00</td>
<td>3.33%</td>
<td>3.88%</td>
</tr>
<tr>
<td>01</td>
<td>3.03%</td>
<td>3.53%</td>
</tr>
<tr>
<td>10</td>
<td>3.7%</td>
<td>4.31%</td>
</tr>
</tbody>
</table>

Note: The data specified in Table 382 and Table 383 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9-bit times when M = 10).
These modes are the following:

- **Mode 0**: Any character starting with a bit at ‘1’.
  In this case the USART measures the duration of the start bit (falling edge to rising edge).

- **Mode 1**: Any character starting with a 10xx bit pattern.
  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.

- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).
  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.

- **Mode 3**: A 0x55 character frame.
  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a ‘0’).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection should be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

**Note:** The BRR value might be corrupted if the USART is disabled (UE=0) during an auto baud rate operation.
53.5.10 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the USART_CR1 register.

**Note:** When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise Mute mode might remain active.

When the Mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to ‘1’. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

**Idle line detection (WAKE=0)**

The USART enters Mute mode when the MMRQ bit is written to ‘1’ and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of Mute mode behavior using Idle line detection is given in Figure 544.
**Figure 544. Mute mode using Idle line detection**

<table>
<thead>
<tr>
<th>RX</th>
<th>Data 1</th>
<th>Data 2</th>
<th>Data 3</th>
<th>Data 4</th>
<th>IDLE</th>
<th>Data 5</th>
<th>Data 6</th>
</tr>
</thead>
</table>

**Note:**

*If the MMRQ is set while the IDLE character has already elapsed, Mute mode is not entered (RWU is not set).*

*If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).*

**4-bit/7-bit address mark detection (WAKE=1)**

In this mode, bytes are recognized as addresses if their MSB is a ‘1’, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

**Note:**

*In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.*

The USART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters Mute mode. When FIFO management is enabled, the software should ensure that there is at least one empty location in the RXFIFO before entering Mute mode.

The USART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

**Note:**

*When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in Mute mode.*

An example of Mute mode behavior using address mark detection is given in Figure 545.
53.5.11 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a Half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

**Modbus/RTU**

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

**Modbus/ASCII**

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE=1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.
53.5.12 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in Table 384.

<table>
<thead>
<tr>
<th>M bits</th>
<th>PCE bit</th>
<th>USART frame(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits are set, the parity bit is equal to 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).
53.5.13 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to Section 53.4: USART implementation on page 1953.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- CLKEN in the USART_CR2 register,
- STOP[1:0], SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in Section 53.5.4 has to be applied for LIN Master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 ‘0’ bits as a break character. Then two bits of value ‘1’ are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE=1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL=1 in USART_CR2) consecutive bits are detected as ‘0’, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1’ is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at ‘0’, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1’, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the Figure 546: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 1979.

Examples of break frames are given on Figure 547: Break detection in LIN mode vs. Framing error detection on page 1980.
Figure 546. Break detection in LIN mode (11-bit break length - LBDL bit is set)

Case 1: break signal not long enough => break discarded, LBDF is not set

<table>
<thead>
<tr>
<th>RX line</th>
<th>Break frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture strobe</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Break state machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
</tr>
</tbody>
</table>

Read samples: 0 0 0 0 0 0 0 0 0 0 1

Case 2: break signal just long enough => break detected, LBDF is set

<table>
<thead>
<tr>
<th>RX line</th>
<th>Break frame</th>
<th>Delimiter is immediate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture strobe</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Break state machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
</tr>
</tbody>
</table>

Read samples: 0 0 0 0 0 0 0 0 0 0 0

Case 3: break signal long enough => break detected, LBDF is set

<table>
<thead>
<tr>
<th>RX line</th>
<th>Break frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture strobe</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Break state machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
</tr>
</tbody>
</table>

Read samples: 0 0 0 0 0 0 0 0 0 0 0

LBDF

MSv31156V1
53.5.14 USART synchronous mode

Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to ‘1’. In synchronous mode, the following bits must be kept cleared:
- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The SCLK pin is the output of the USART transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see Figure 548, Figure 549 and Figure 550).

During the Idle state, preamble and send break, the external SCLK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since SCLK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on SCLK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).
Note: In master mode, the SCLK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 548. USART example of synchronous master transmission

Figure 549. USART data clock timing diagram in synchronous master mode (M bits =00)
Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to ‘1’. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The SCLK pin is the input of the USART in slave mode.

Note: *When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (usart_ker_ck_pres) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see Figure 551).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.
Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- **Software NSS management (DIS_NSS = 1)**
  - The SPI slave is always selected and NSS input pin is ignored.
  - The external NSS pin remains free for other application uses.

- **Hardware NSS management (DIS_NSS = 0)**
  - The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

*Note:* The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE=0) to ensure that the clock pulses function correctly. In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

### SPI Slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.
In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first SCLK transmission edge. To avoid this underrun error, the USART_TDR should be written 3 usart_ker_ck cycles before the first SCLK edge.

53.5.15 USART single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to ‘1’:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

53.5.16 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = ‘00’ or STOP = ‘11’
- from the end of the second stop bit if STOP = ‘10’.
- from the beginning of the stop bit if STOP = ‘01’.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.
53.5.17 USART Smartcard mode

This section is relevant only when Smartcard mode is supported. Refer to Section 53.4: USART implementation on page 1953.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In Smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the Smartcard.

The Smartcard interface is designed to support asynchronous Smartcard protocol as defined in the ISO 7816-3 standard. Both T=0 (character mode) and T=1 (block mode) are supported.

The USART should be configured as:

- 8 bits plus parity: M=1 and PCE=1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP=’11’ in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T=0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 552 shows examples of what can be seen on the data line with and without parity error.

![Figure 552. ISO 7816-3 asynchronous protocol](image)

When connected to a Smartcard, the TX output of the USART drives a bidirectional line that is also driven by the Smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.

- In transmission, if the Smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.
The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.

- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T=1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the Smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.

- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn’t repeat the character.

- In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).

- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.

- The deassertion of TC flag is unaffected by Smartcard mode.

- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.

- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: Break characters are not significant in Smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No idle frame is transmitted when toggling the TE bit. The idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 553 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.
The USART can provide a clock to the Smartcard through the SCLK output. In Smartcard mode, SCLK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. SCLK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

**Block mode (T=1)**

In T=1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the Smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

**Note:** The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the Smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time -11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the Smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

**Note:** As in the Smartcard protocol definition, the BWT/CWT values should be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT - 11 or CWT -11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the Smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value.
(0x0). With this value, an interrupt is generated after the 4th received character. The 
software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by 
programming the BLEN value. However, before the start of the block, the maximum value of 
BLEN (0xFF) may be programmed. The real value is programmed after the reception of the 
third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the 
BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must 
be programmed. The total block length (including prologue, epilogue and information fields) 
equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and 
interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt 
(Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The Smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state 
of the line and parity is even. In order to use this convention, the following control bits must 
be programmed: MSBFIRST=0, DATAINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state 
on the signal line and parity is even. In order to use this convention, the following control bits 
must be programmed: MSBFIRST=1, DATAINV=1.

Note: When logical data values are inverted (0=H, 1=L), the parity bit is also inverted in the same 
way.

In order to recognize the card convention, the card sends the initial character, TS, as the 
first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS 
are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and 
  moment 2 conveys the most significant bit (MSB first). When decoded by inverse 
  convention, the conveyed byte is equal to ‘3F’.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and 
  moment 2 conveys the least significant bit (LSB first). When decoded by direct 
  convention, the conveyed byte is equal to ‘3B’.

Character parity is correct when there is an even number of bits set to 1 in the nine 
moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to 
recognize either pattern and act accordingly. The pattern recognition is not done in 
hardware, but through a software sequence. Moreover, supposing that the USART is 
configured in direct convention (default) and the card answers with the inverse convention, 
TS = LHHL LLL LLH => the USART received character is 03 and the parity is odd.
Therefore, two methods are available for TS pattern recognition:

**Method 1**

The USART is programmed in standard Smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention.
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART.

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

**Method 2**

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

- (H) LHHL LLL LLH = 0x103 -> inverse convention to be chosen
- (H) LHHL HHH LLH = 0x13B -> direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

### 53.5.18 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to Section 53.4: USART implementation on page 1953.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see Figure 554).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 Kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not
encoded. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.

- A ‘0’ is transmitted as a high pulse and a ‘1’ is transmitted as a ‘0’. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see Figure 555).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder doesn’t work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to ‘1 stop bit’.

IrDA low-power mode

- Transmitter
  In low-power mode, the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < PSC< 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver
  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in the USART_GTPR).

Note: A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.

The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).
Figure 554. IrDA SIR ENDEC block diagram

Figure 555. IrDA data modulation (3/16) - Normal mode
53.5.19 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

*Note:* Refer to Section 53.4: USART implementation on page 1953 to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 53.5.6. To perform continuous communications when the FIFO is disabled, clear the TXE/RXNE flags in the USART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding Direct memory access controller section) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.
Figure 556. Transmission using DMA

<table>
<thead>
<tr>
<th>TX line</th>
<th>Frame 1</th>
<th>Frame 2</th>
<th>Frame 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle preamble</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TXE flag</th>
<th>DMA request</th>
<th>USART_TDR</th>
<th>TC flag</th>
<th>DMA writes USART_TDR</th>
<th>DMA TCIF flag (transfer complete)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set by hardware cleared by DMA read</td>
<td>Set by hardware cleared by DMA read</td>
<td>F1</td>
<td>Cleared by software</td>
<td>DMA writes F1 into USART_TDR</td>
<td>Software configures DMA to send 3 data blocks and enables USART</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F2</td>
<td></td>
<td>DMA writes F2 into USART_TDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>F3</td>
<td></td>
<td>DMA writes F3 into USART_TDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The DMA transfer is complete (TCIF=1 in DMA_ISR)</td>
<td></td>
</tr>
</tbody>
</table>

When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

**Reception using DMA**

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to the corresponding Direct memory access controller section) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register.
5. Configure interrupt generation after half/full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.
Figure 557. Reception using DMA

Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

53.5.20 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The Figure 558 shows how to connect 2 devices in this mode:

Figure 558. Hardware flow control between 2 USARTs
RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to ‘1’ in the USART_CR3 register.

**RS232 RTS flow control**

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 559* shows an example of communication with RTS flow control enabled.

**Figure 559. RS232 RTS flow control**

![RS232 RTS flow control diagram](MSv31168V2)

*Note:* When FIFO mode is enabled, nRTS is deasserted only when RXFIFO is full.

**RS232 CTS flow control**

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 560* shows an example of communication with CTS flow control enabled.
For correct behavior, nCTS must be asserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the USART_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).
53.5.21 **USART low-power management**

The USART has advanced low-power mode functions, that enables transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the USART provides a wakeup interrupt (usart_wkup) if a specific action requiring the activation of the usart_pclk clock is needed:

- **If FIFO mode is disabled**
  - usart_pclk clock has to be activated to empty the USART data register.
  - In this case, the usart_wkup interrupt source is RXNE set to ‘1’. The RXNEIE bit must be set before entering low-power mode.

- **If FIFO mode is enabled**
  - usart_pclk clock has to be activated to:
    - to fill the TXFIFO
    - or to empty the RXFIFO
  - In this case, the usart_wkup interrupt source can be:
    - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
    - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
    - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart_wkup interrupt source can be one of the following events:

- TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
- RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

For example, the application can set the threshold to the maximum RXFIFO size if the wakeup time is less than the time required to receive a single byte across the line.

Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wakeup the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific usart_wkup interrupt can be selected through the WUS bitfields.

When the wakeup event is detected, the WUF flag is set by hardware and a usart_wkup interrupt is generated if the WUFIE bit is set. In this case the usart_wkup interrupt is not mandatory and setting the WUF being is sufficient to wake up the MCU from low-power mode.
Before entering low-power mode, make sure that no USART transfers are ongoing.
Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in low-power or active mode.
When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.
When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled when exiting from low-power mode.
When the FIFO is enabled, waking up from low-power mode on address match is only possible when Mute mode is enabled.

Using Mute mode with low-power mode

If the USART is put into Mute mode before entering low-power mode:
• Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
• If the wakeup from Mute mode on address match is used, then the low-power mode wakeup source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in Mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, Mute mode can be used with wakeup from low-power mode without any constraints (i.e., the two points mentioned above about Mute and low-power mode are valid only when FIFO management is disabled).

Wakeup from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.
If the wakeup event is verified, the MCU wakes up from low-power mode and data reception goes on normally.
If the wakeup event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.
The example below shows the case of a wakeup event programmed to “address match detection” and FIFO management disabled.
Figure 561 shows the USART behavior when the wakeup event is verified.

**Figure 561. Wakeup event verified (wakeup event = address match, FIFO disabled)**

![Diagram showing USART behavior when wakeup event is verified](MSv40856V2)

Figure 562 shows the USART behavior when the wakeup event is not verified.

**Figure 562. Wakeup event not verified (wakeup event = address match, FIFO disabled)**

![Diagram showing USART behavior when wakeup event is not verified](MSv40857V2)

**Note:** The figures above are valid when address match or any received frame is used as wakeup event. If the wakeup event is the start bit detection, the USART sends the wakeup event to the MCU at the end of the start bit.
Determining the maximum USART baud rate that enables to correctly wake up the microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wakeup time parameter (refer to the device datasheet) and on the USART receiver tolerance (see Section 53.5.8: Tolerance of the USART receiver to clock deviation).

Let us take the example of OVER8 = 0, M bits = '01', ONEBIT = 0 and BRR [3:0] = 0000. In these conditions, according to Table 382: Tolerance of the USART receiver when BRR [3:0] = 0000, the USART receiver tolerance equals 3.41%.

\[ D_{\text{WUmax}} = \frac{t_{\text{WUUSART}}}{11 \times T_{\text{bit Min}}} \]
\[ T_{\text{bit Min}} = \frac{t_{\text{WUUSART}}}{11 \times D_{\text{WUmax}}} \]

where \( t_{\text{WUUSART}} \) is the wakeup time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the \text{usart_ker_ck} inaccuracy.

For example, if HSI is used as \text{usart_ker_ck}, and the HSI inaccuracy is of 1%, then we obtain:

\[ t_{\text{WUUSART}} = 3 \, \mu s \] (values provided only as examples; for correct values, refer to the device datasheet).
\[ D_{\text{WUmax}} = 3.41\% - 1\% = 2.41\% \]
\[ T_{\text{bit min}} = 3 \, \mu s / (11 \times 2.41\%) = 11.32 \, \mu s. \]

As a result, the maximum baud rate that enables to wakeup correctly from low-power mode is: \(1/11.32 \, \mu s = 88.36 \, \text{Kbaud}\).

53.6 USART interrupts

During USART communications, an interrupt (\text{usart_it}) can be generated by different events. The USART block can also generate a wakeup interrupt (\text{usart_wkup}).

Refer to Table 385 for a detailed description of all USART interrupt requests.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Interrupt activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit data register empty</td>
<td>TXE</td>
<td>TXEIE</td>
<td>TXE cleared when a data is written in TDR</td>
<td>YES</td>
</tr>
<tr>
<td>Transmit FIFO Not Full</td>
<td>TXFNF</td>
<td>TXFNFIE</td>
<td>TXFNF cleared when TXFIFO is full.</td>
<td>YES</td>
</tr>
<tr>
<td>Transmit FIFO Empty</td>
<td>TXFE</td>
<td>TXFEIE</td>
<td>TXFE cleared when the TXFIFO contains at least one data or by setting TXFRQ bit.</td>
<td>YES</td>
</tr>
</tbody>
</table>

Table 385. USART interrupt requests
### Table 385. USART interrupt requests (continued)

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Interrupt activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold</td>
<td>YES YES</td>
</tr>
<tr>
<td>CTS interrupt</td>
<td>CTSIE</td>
<td>CTSIF cleared by software by setting CTSCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Transmission Complete</td>
<td>TC</td>
<td>TC cleared when a data is written in TDR or by setting TCCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Transmission Complete Before Guard Time</td>
<td>TCBGT</td>
<td>TCBGT cleared when a data is written in TDR or by setting TCBGTCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Receive data register not empty (data ready to be read)</td>
<td>RXNE</td>
<td>RXNE cleared by reading RDR or by setting RXFRQ bit.</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO Not Empty</td>
<td>RXFNE</td>
<td>RXFNE cleared when the RXFIFO is empty or by setting RXFRQ bit.</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO Full</td>
<td>RXFF(1)</td>
<td>RXFF is cleared when the RXFIFO contains at least one data.</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold</td>
<td>YES YES</td>
</tr>
<tr>
<td>Overrun error detected</td>
<td>ORE</td>
<td>ORE cleared by setting ORECF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Idle line detected</td>
<td>IDLE</td>
<td>IDLE cleared by setting IDLECF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Parity error</td>
<td>PE</td>
<td>PE cleared by setting PECF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>LIN break</td>
<td>LBDF</td>
<td>LBDF cleared by setting LBDCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Noise error, Overrun error and Framing Error in multibuffer communication</td>
<td>NE or ORE or FE</td>
<td>NE cleared by setting NECF bit. ORE cleared by setting ORECF bit. FE flag cleared by setting FECF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Character match</td>
<td>CMF</td>
<td>CMF cleared by setting CMCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Receiver timeout</td>
<td>RTOF</td>
<td>RTOF cleared by setting RTOCCF bit.</td>
<td>YES NO</td>
</tr>
</tbody>
</table>
Table 385. USART interrupt requests (continued)

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Interrupt activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>End of Block</td>
<td>EOBF</td>
<td>EOBIE</td>
<td>EOBF is cleared by setting EOBCF bit.</td>
<td>YES</td>
</tr>
<tr>
<td>Wakeup from low-power mode</td>
<td>WUF(2)</td>
<td>WUFIE</td>
<td>WUF is cleared by setting WUCF bit.</td>
<td>NO</td>
</tr>
<tr>
<td>SPI slave underrun error</td>
<td>UDR</td>
<td>EIE</td>
<td>UDR is cleared by setting UDRCF bit.</td>
<td>YES</td>
</tr>
<tr>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFTIE</td>
<td>TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold</td>
<td>YES</td>
</tr>
<tr>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFTIE</td>
<td>RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold</td>
<td>YES</td>
</tr>
</tbody>
</table>

1. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).

2. The WUF interrupt is active only in low-power mode.
53.7 **USART registers**

Refer to Section 1.2 on page 97 for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

53.7.1 **USART control register 1 [alternate] (USART_CR1)**

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

### FIFO mode enabled

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXFFIE</td>
<td>TXFEIE</td>
<td>FIFOEN</td>
<td>M1</td>
<td>EOBIE</td>
<td>RTOIE</td>
<td>DEAT[4:0]</td>
<td>DEDT[4:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>OVER8</td>
<td>CMIE</td>
<td>MME</td>
<td>M0</td>
<td>WAKE</td>
<td>PCE</td>
<td>PS</td>
<td>PEIE</td>
<td>TXFNFIE</td>
<td>TCIE</td>
<td>RXFNEIE</td>
<td>IDLEIE</td>
<td>TE</td>
<td>RE</td>
<td>UESM</td>
<td>UE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bit 31 RXFFIE**: RXFIFO Full interrupt enable
   This bit is set and cleared by software.
   - 0: Interrupt inhibited
   - 1: USART interrupt generated when RXFF=1 in the USART_ISR register

**Bit 30 TXFEIE**: TXFIFO empty interrupt enable
   This bit is set and cleared by software.
   - 0: Interrupt inhibited
   - 1: USART interrupt generated when TXFE=1 in the USART_ISR register

**Bit 29 FIFOEN**: FIFO mode enable
   This bit is set and cleared by software.
   - 0: FIFO mode is disabled.
   - 1: FIFO mode is enabled.
   This bitfield can only be written when the USART is disabled (UE=0).
   Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in Smartcard modes only. It must not be enabled in IrDA and LIN modes.

**Bit 28 M1**: Word length
   This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
   \[M[1:0] = '00': 1\text{ start bit, 8 Data bits, n Stop bit}\]
   \[M[1:0] = '01': 1\text{ start bit, 9 Data bits, n Stop bit}\]
   \[M[1:0] = '10': 1\text{ start bit, 7 Data bits, n Stop bit}\]
   This bit can only be written when the USART is disabled (UE=0).
   Note: In 7-bits data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.
Bit 27 EOBIE: End of Block interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the EOBF flag is set in the USART_ISR register
Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bit 26 RTOIE: Receiver timeout interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.
Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 53.4: USART implementation on page 1953.

Bits 25:21 DEAT[4:0]: Driver Enable assertion time
This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).
This bitfield can only be written when the USART is disabled (UE=0).
Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bits 20:16 DEDT[4:0]: Driver Enable deassertion time
This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).
If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.
This bitfield can only be written when the USART is disabled (UE=0).
Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bit 15 OVER8: Oversampling mode
0: Oversampling by 16
1: Oversampling by 8
This bit can only be written when the USART is disabled (UE=0).
Note: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.

Bit 14 CMIE: Character match interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 MME: Mute mode enable
This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.
0: Receiver in active mode permanently
1: Receiver can switch between Mute mode and active mode.

Bit 12 M0: Word length
This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).
This bit can only be written when the USART is disabled (UE=0).
Bit 11 **WAKE**: Receiver wakeup method  
This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.  
0: Idle line  
1: Address mark  
This bitfield can only be written when the USART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable  
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).  
0: Parity control disabled  
1: Parity control enabled  
This bitfield can only be written when the USART is disabled (UE=0).

Bit 9 **PS**: Parity selection  
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.  
0: Even parity  
1: Odd parity  
This bitfield can only be written when the USART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever PE=1 in the USART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever TXFN=1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever TC=1 in the USART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever ORE=1 or RXFNE=1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever IDLE=1 in the USART_ISR register
Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled
1: Transmitter is enabled

*Note*: During transmission, a low pulse on the TE bit (‘0’ followed by ‘1’) sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to ‘1’. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode. When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.
1: USART able to wake up the MCU from low-power mode.

*Note*: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.

*If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode
1: USART enabled

*Note*: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In Smartcard mode, (SCEN = 1), the SCLK is always available when CLKEN = 1, regardless of the UE bit value.
53.7.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00
Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

**FIFO mode disabled**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>FIFOEN</td>
<td>FIFO mode enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: FIFO mode is disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: FIFO mode is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bitfield can only be written when the USART is disabled (UE=0).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> FIFO mode can be used on standard UART communication, in SPI master/slave mode and in Smartcard modes only. It must not be enabled in IrDA and LIN modes.</td>
</tr>
<tr>
<td>28</td>
<td>M1</td>
<td>Word length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M[1:0] = '00': 1 start bit, 8 Data bits, n Stop bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M[1:0] = '01': 1 start bit, 9 Data bits, n Stop bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M[1:0] = '10': 1 start bit, 7 Data bits, n Stop bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit can only be written when the USART is disabled (UE=0).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> In 7-bits data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.</td>
</tr>
<tr>
<td>27</td>
<td>EOBIE</td>
<td>End of Block interrupt enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Interrupt inhibited</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: USART interrupt generated when the EOBF flag is set in the USART_ISR register</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.</td>
</tr>
<tr>
<td>26</td>
<td>RTOIE</td>
<td>Receiver timeout interrupt enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Interrupt inhibited</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: USART interrupt generated when the RTOF bit is set in the USART_ISR register</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 53.4: USART implementation on page 1953.</td>
</tr>
</tbody>
</table>
Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE=0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE=0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16
1: Oversampling by 8

This bit can only be written when the USART is disabled (UE=0).

*Note: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.*

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited
1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently
1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the USART is disabled (UE=0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line
1: Address mark

This bitfield can only be written when the USART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software.

Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled
1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE=0).
Bit 9  **PS**: Parity selection  
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.  
0: Even parity  
1: Odd parity  
This bitfield can only be written when the USART is disabled (UE=0).

Bit 8  **PEIE**: PE interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever PE=1 in the USART_ISR register

Bit 7  **TxEIE**: Transmit data register empty  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever TXE =1 in the USART_ISR register

Bit 6  **TCIE**: Transmission complete interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever TC=1 in the USART_ISR register

Bit 5  **RXNEIE**: Receive data register not empty  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever ORE=1 or RXNE=1 in the USART_ISR register

Bit 4  **IDLEIE**: IDLE interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever IDLE=1 in the USART_ISR register

Bit 3  **TE**: Transmitter enable  
This bit enables the transmitter. It is set and cleared by software.  
0: Transmitter is disabled  
1: Transmitter is enabled

**Note:** During transmission, a low pulse on the TE bit ('0' followed by '1') sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1'. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.  
In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.
Bit 2 **RE**: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode
When this bit is cleared, the USART cannot wake up the MCU from low-power mode.
When this bit is set, the USART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: USART not able to wake up the MCU from low-power mode.
1: USART able to wake up the MCU from low-power mode.

*Note:* It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.
If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bit 0 **UE**: USART enable
When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.
0: USART prescaler and outputs disabled, low-power mode
1: USART enabled

*Note:* To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.
In Smartcard mode, (SCEN = 1), the SCLK is always available when CLKEN = 1, regardless of the UE bit value.

### 53.7.3 USART control register 2 (USART_CR2)

Address offset: 0x04
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWAP</td>
<td>LINEN</td>
<td>STOP[1:0]</td>
<td>CLKEN</td>
<td>CPOL</td>
<td>CPHA</td>
<td>LBCL</td>
<td>Res.</td>
<td>LBDIE</td>
<td>LBDL</td>
<td>ADDM7</td>
<td>Res.</td>
<td>Res.</td>
<td>SLVEN</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:24  ADD[7:0]: Address of the USART node

ADD[7:4]:
These bits give the address of the USART node or a character code to be recognized.
They are used to wake up the MCU with 7-bit address mark detection in multiprocessor
communication during Mute mode or low-power mode. The MSB of the character sent by the
transmitter should be equal to 1. They can also be used for character detection during normal
reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this
case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on
match.
These bits can only be written when reception is disabled (RE = 0) or the USART is disabled
(UE=0).

ADD[3:0]:
These bits give the address of the USART node or a character code to be recognized.
They are used for wakeup with address mark detection, in multiprocessor communication during
Mute mode or low-power mode.
These bits can only be written when reception is disabled (RE = 0) or the USART is disabled
(UE=0).

Bit 23  RTOEN: Receiver timeout enable
This bit is set and cleared by software.
0: Receiver timeout feature disabled.
1: Receiver timeout feature enabled.
When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle
(no reception) for the duration programmed in the RTOR (receiver timeout register).
Note:  If the USART does not support the Receiver timeout feature, this bit is reserved and must be
kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bits 22:21  ABRMOD[1:0]: Auto baud rate mode
These bits are set and cleared by software.
00: Measurement of the start bit is used to detect the baud rate.
01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 ->
Frame = Start10xxxxxx)
10: 0x7F frame detection.
11: 0x55 frame detection
This bitfield can only be written when ABREN = 0 or the USART is disabled (UE=0).
Note:  If DATAINV=1 and/or MSBFIRST=1 the patterns must be the same on the line, for example
0xAA for MSBFIRST)
If the USART does not support the auto baud rate feature, this bit is reserved and must be kept
at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bit 20  ABREN: Auto baud rate enable
This bit is set and cleared by software.
0: Auto baud rate detection is disabled.
1: Auto baud rate detection is enabled.
Note:  If the USART does not support the auto baud rate feature, this bit is reserved and must be kept
at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bit 19  MSBFIRST: Most significant bit first
This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bitfield can only be written when the USART is disabled (UE=0).
Bit 18 DATAINV: Binary data inversion
   This bit is set and cleared by software.
   0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
   1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The
   parity bit is also inverted.
   This bitfield can only be written when the USART is disabled (UE=0).

Bit 17 TXINV: TX pin active level inversion
   This bit is set and cleared by software.
   0: TX pin signal works using the standard logic levels (VDD =1/idle, Gnd=0/mark)
   1: TX pin signal values are inverted. ((VDD =0/mark, Gnd=1/idle).
   This enables the use of an external inverter on the TX line.
   This bitfield can only be written when the USART is disabled (UE=0).

Bit 16 RXINV: RX pin active level inversion
   This bit is set and cleared by software.
   0: RX pin signal works using the standard logic levels (VDD =1/idle, Gnd=0/mark)
   1: RX pin signal values are inverted. ((VDD =0/mark, Gnd=1/idle).
   This enables the use of an external inverter on the RX line.
   This bitfield can only be written when the USART is disabled (UE=0).

Bit 15 SWAP: Swap TX/RX pins
   This bit is set and cleared by software.
   0: TX/RX pins are used as defined in standard pinout
   1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired
   connection to another UART.
   This bitfield can only be written when the USART is disabled (UE=0).

Bit 14 LINEN: LIN mode enable
   This bit is set and cleared by software.
   0: LIN mode disabled
   1: LIN mode enabled
   The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the
   SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.
   This bitfield can only be written when the USART is disabled (UE=0).
   Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value.
   Refer to Section 53.4: USART implementation on page 1953.

Bits 13:12 STOP[1:0]: stop bits
   These bits are used for programming the stop bits.
   00: 1 stop bit
   01: 0.5 stop bit.
   10: 2 stop bits
   11: 1.5 stop bits
   This bitfield can only be written when the USART is disabled (UE=0).
Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the SCLK pin.

- 0: SCLK pin disabled
- 1: SCLK pin enabled

This bit can only be written when the USART is disabled (UE=0).

**Note:** If neither synchronous mode nor Smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

In Smartcard mode, in order to provide correctly the SCLK clock to the smartcard, the steps below must be respected:

- **UE = 0**
- **SCEN = 1**
- GTPR configuration
- **CLKEN = 1**
- **UE = 1**

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship.

- 0: Steady low value on SCLK pin outside transmission window
- 1: Steady high value on SCLK pin outside transmission window

This bit can only be written when the USART is disabled (UE=0).

**Note:** If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see Figure 542 and Figure 543).

- 0: The first clock transition is the first data capture edge
- 1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE=0).

**Note:** If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin in synchronous mode.

- 0: The clock pulse of the last data bit is not output to the SCLK pin
- 1: The clock pulse of the last data bit is output to the SCLK pin

**Caution:** The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE=0).

**Note:** If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

- 0: Interrupt is inhibited
- 1: An interrupt is generated whenever LBDF=1 in the USART_ISR register

**Note:** If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.
Bit 5  **LBDL**: LIN break detection length  
   This bit is for selection between 11 bit or 10 bit break detection.  
   0: 10-bit break detection  
   1: 11-bit break detection  
   This bit can only be written when the USART is disabled (UE=0).  
   **Note:** If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bit 4  **ADDM7**: 7-bit Address Detection/4-bit Address Detection  
   This bit is for selection between 4-bit address detection or 7-bit address detection.  
   0: 4-bit address detection  
   1: 7-bit address detection (in 8-bit data mode)  
   This bit can only be written when the USART is disabled (UE=0)  
   **Note:** In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3  **DIS_NSS**:  
   When the DIS_NSS bit is set, the NSS pin input is ignored.  
   0: SPI slave selection depends on NSS input pin.  
   1: SPI slave is always selected and NSS input pin is ignored.  
   **Note:** When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bits 2:1  Reserved, must be kept at reset value.

Bit 0  **SLVEN**: Synchronous Slave mode enable  
   When the SLVEN bit is set, the synchronous slave mode is enabled.  
   0: Slave mode disabled.  
   1: Slave mode enabled.  
   **Note:** When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

**Note:** The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.

### 53.7.4 USART control register 3 (USART_CR3)

**Address offset:** 0x08  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEP</td>
<td>DEM</td>
<td>DDRE</td>
<td>OVR</td>
<td>ONE BIT</td>
<td>CTSIE</td>
<td>CTSE</td>
<td>RTSE</td>
<td>DMAT</td>
<td>DMAR</td>
<td>SCEN</td>
<td>NACK</td>
<td>HD SEL</td>
<td>IRLP</td>
<td>IREN</td>
<td>EIE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:29 **TXFTCFG[2:0]**: TXFIFO threshold configuration
- 000: TXFIFO reaches 1/8 of its depth
- 001: TXFIFO reaches 1/4 of its depth
- 010: TXFIFO reaches 1/2 of its depth
- 011: TXFIFO reaches 3/4 of its depth
- 100: TXFIFO reaches 7/8 of its depth
- 101: TXFIFO becomes empty
- Remaining combinations: Reserved

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]**: Receive FIFO threshold configuration
- 000: Receive FIFO reaches 1/8 of its depth
- 001: Receive FIFO reaches 1/4 of its depth
- 010: Receive FIFO reaches 1/2 of its depth
- 011: Receive FIFO reaches 3/4 of its depth
- 100: Receive FIFO reaches 7/8 of its depth
- 101: Receive FIFO becomes full
- Remaining combinations: Reserved

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated whenever TCBGT=1 in the USART_ISR register

*Note: If the USART does not support the Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wakeup from low-power mode interrupt enable
- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated whenever WUF=1 in the USART_ISR register

*Note: **WUFIE must be set before entering in low-power mode.**

*The WUF interrupt is active only in low-power mode.*

*If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*
Bits 21:20  **WUS[1:0]:** Wakeup from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (Wakeup from low-power mode flag).

- **00:** WUF active on address match (as defined by ADD[7:0] and ADDM7)
- **01:** Reserved.
- **10:** WUF active on start bit detection
- **11:** WUF active on RXNE/RXFNE.

This bitfield can only be written when the USART is disabled (UE=0).

*If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bits 19:17  **SCARCTN[2:0]:** Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in Smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number of erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE=0).

When the USART is enabled (UE=1), this bitfield may only be written to 0x0, in order to stop retransmission.

- **0x0:** retransmission disabled - No automatic retransmission in transmit mode.
- **0x1** to **0x7:** number of automatic retransmission attempts (before signaling error)

*Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 16  Reserved, must be kept at reset value.

Bit 15  **DEP:** Driver enable polarity selection

- **0:** DE signal is active high.
- **1:** DE signal is active low.

This bit can only be written when the USART is disabled (UE=0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 14  **DEM:** Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

- **0:** DE function is disabled.
- **1:** DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE=0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 53.4: USART implementation on page 1953.*

Bit 13  **DDRE:** DMA Disable on Reception Error

- **0:** DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred (used for Smartcard mode).
- **1:** DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE/RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE=0).

*Note: The reception errors are: parity error, framing error or noise error.*
Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data is written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.
This bit can only be written when the USART is disabled (UE=0).
*Note: This control bit enables checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable
This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.
0: Three sample bit method
1: One sample bit method
This bit can only be written when the USART is disabled (UE=0).

Bit 10 **CTSE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF=1 in the USART_ISR register
*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.
This bit can only be written when the USART is disabled (UE=0)
*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.
This bit can only be written when the USART is disabled (UE=0).
*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception
Bit 5  **SCEN**: Smartcard mode enable
This bit is used for enabling Smartcard mode.
0: Smartcard Mode disabled
1: Smartcard Mode enabled
This bitfield can only be written when the USART is disabled (UE=0).
*Note*: **If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.**

Bit 4  **NACK**: Smartcard NACK enable
0: NACK transmission in case of parity error is disabled
1: NACK transmission during parity error is enabled
This bitfield can only be written when the USART is disabled (UE=0).
*Note*: **If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.**

Bit 3  **HDSEL**: Half-duplex selection
Selection of Single-wire Half-duplex mode
0: Half duplex mode is not selected
1: Half duplex mode is selected
This bit can only be written when the USART is disabled (UE=0).

Bit 2  **IRLP**: IrDA low-power
This bit is used for selecting between normal and low-power IrDA modes
0: Normal mode
1: Low-power mode
This bit can only be written when the USART is disabled (UE=0).
*Note*: **If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.**

Bit 1  **IREN**: IrDA mode enable
This bit is set and cleared by software.
0: IrDA disabled
1: IrDA enabled
This bit can only be written when the USART is disabled (UE=0).
*Note*: **If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.**

Bit 0  **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE=1 or ORE=1 or NE=1 or UDR = 1 in the USART_ISR register).
0: Interrupt inhibited
1: interrupt generated when FE=1 or ORE=1 or NE=1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.
53.7.5  USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15:0</td>
<td>BRR[15:0]: USART baud rate</td>
</tr>
</tbody>
</table>

BRR[15:4]


BRR[3:0]

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].
When OVER8 = 1:
BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
BRR[3] must be kept cleared.

53.7.6  USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15:0</td>
<td>GT[7:0] PSC[7:0]</td>
</tr>
</tbody>
</table>

GT[7:0]
PSC[7:0]
53.7.7 USART receiver timeout register (USART_RTOR)

Address offset: 0x14
Reset value: 0x0000 0000

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 GT[7:0]: Guard time value
This bitfield is used to program the Guard time value in terms of number of baud clock periods.
This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.
This bitfield can only be written when the USART is disabled (UE=0).
Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value.
Refer to Section 53.4: USART implementation on page 1953.

Bits 7:0 PSC[7:0]: Prescaler value
In IrDA low-power and normal IrDA mode:
PSC[7:0] = IrDA Normal and Low-Power baud rate
PSC[7:0] is used to program the prescaler for dividing the USART source clock to achieve the low-power frequency: the source clock is divided by the value given in the register (8 significant bits):
In Smartcard mode:
PSC[4:0] = Prescaler value
PSC[4:0] is used to program the prescaler for dividing the USART source clock to provide the Smartcard clock. The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value
00001: Divides the source clock by 1 (IrDA mode)/by 2 (Smartcard mode)
00010: Divides the source clock by 2 (IrDA mode)/by 4 (Smartcard mode)
00011: Divides the source clock by 3 (IrDA mode)/by 6 (Smartcard mode)
... 11111: Divides the source clock by 15 (IrDA mode)/by 30 (Smartcard mode)

This bitfield can only be written when the USART is disabled (UE=0).
Note: Bits [7:5] must be kept cleared if Smartcard mode is used.
This bitfield is reserved and forced by hardware to '0' when the Smartcard and IrDA modes are not supported. Refer to Section 53.4: USART implementation on page 1953.

2020/2974 RM0455 Rev 3
Bits 31:24 **BLEN[7:0]**: Block Length

This bitfield gives the Block length in Smartcard T=1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:
- BLEN = 0 -> 0 information characters + LEC
- BLEN = 1 -> 0 information characters + CRC
- BLEN = 255 -> 254 information characters + CRC (total 256 characters)

In Smartcard mode, the Block length counter is reset when TXE=0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the Block length counter is reset when RE=0 (receiver disabled) and/or when the EOBCF bit is written to 1.

*Note:* This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bits during which there is no activity on the RX line.

In standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In Smartcard mode, this value is used to implement the CWT and BWT. See Smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

*Note:* This value must only be programmed once per received character.

*Note:* RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to “0x00000000” when the Receiver timeout feature is not supported. Refer to Section 53.4: USART implementation on page 1953.

### 53.7.8 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>
Universal synchronous/asynchronous receiver transmitter (USART/UART)  RM0455

53.7.9 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C
Reset value: 0x0080 00C0

X = 2 if FIFO/Smartcard mode is enabled
X = 0 if FIFO is enabled and Smartcard mode is disabled

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

**FIFO mode enabled**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ABRF</td>
<td>ABRE</td>
<td>UDR</td>
<td>EOFB</td>
<td>RTOF</td>
<td>CTS</td>
<td>CTSIF</td>
<td>LBDF</td>
<td>TXFNF</td>
<td>TC</td>
<td>RXFNE</td>
<td>IDLE</td>
<td>ORE</td>
<td>NE</td>
<td>FE</td>
<td>PE</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**Bits 31:5** Reserved, must be kept at reset value.

**Bit 4 TXFREQ:** Transmit data flush request

When FIFO mode is disabled, writing ‘1’ to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in Smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFREQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and Smartcard modes.

*Note:* In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

**Bit 3 RXFREQ:** Receive data flush request

Writing 1 to this bit empties the entire receive FIFO i.e. clears the bit RXFNE. This enables to discard the received data without reading them, and avoid an overrun condition.

**Bit 2 MMRQ:** Mute mode request

Writing 1 to this bit puts the USART in Mute mode and resets the RWU flag.

**Bit 1 SBKRQ:** Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

*Note:* When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.

**Bit 0 ABRRQ:** Auto baud rate request

Writing 1 to this bit resets the ABRF flag in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

*Note:* If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.
Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag
This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG of USART_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit =1 (bit 31) in the USART_CR3 register.
0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag
This bit is set by hardware when the threshold programmed in RXFTCFG in USART_CR3 register is reached. This means that there are (RXFTCFG - 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit =1 (bit 27) in the USART_CR3 register.
0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG threshold is configured to ‘101’, RXFT flag is set if 16 data are available i.e. 15 data in the RXFIFO and 1 data in the USART_RDR. Consequently, the 17th received data does not cause an overrun error. The overrun error occurs after receiving the 18th data.

Bit 25 **TCBGT**: Transmission complete before guard time flag
This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.
It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE=1 in the USART_CR3 register.
This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)
1: Transmission is complete successfully (before Guard time completion and there is no NACK from the Smart card).

Note: If the USART does not support the Smartcard mode, this bit is reserved and kept at reset value. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is ‘1’. Refer to Section 53.4: USART implementation on page 1953.

Bit 24 **RXFF**: RXFIFO full
This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.
An interrupt is generated if the RXFFIE bit =1 in the USART_CR1 register.
0: RXFIFO not full.
1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO empty
This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.
An interrupt is generated if the TXFEIE bit =1 (bit 30) in the USART_CR1 register.
0: TXFIFO not empty.
1: TXFIFO empty.
Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

**Note**: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE=1 in the USART_CR3 register.

**Note**: When UESM is cleared, WUF flag is also cleared.

The WUF interrupt is active only in low-power mode.

If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bit 19 **RWU**: Receiver wakeup from Mute mode

This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode
1: Receiver in Mute mode

**Note**: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted
1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE=1 in the USART_CR1 register.

0: No Character match detected
1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)
1: Reception on going
Bit 15 **ABRF**: Auto baud rate flag
This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXFNE and FE are also set in this case). It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 14 **ABRE**: Auto baud rate error
This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed). It is cleared by software, by writing 1 to the ABRRQ bit in the USART_CR3 register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 13 **UDR**: SPI slave underrun error flag
In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error
1: underrun error

*Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 12 **EOBF**: End of block flag
This bit is set by hardware when a complete block has been received (for example T=1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4. An interrupt is generated if the EOBIE=1 in the USART_CR2 register. It is cleared by software, writing 1 to the EOBCF in the USART_ICR register.

0: End of Block not reached
1: End of Block (number of characters) reached

*Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 11 **RTOF**: Receiver timeout
This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE=1 in the USART_CR2 register. In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached
1: Timeout value reached without any data reception

*Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set. The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.*

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.*
Bit 10 **CTS**: CTS flag
This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.
0: nCTS line set
1: nCTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag
This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE=1 in the USART_CR3 register.
0: No change occurred on the nCTS status line
1: A change occurred on the nCTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 **LBDF**: LIN break detection flag
This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.
0: LIN Break not detected
1: LIN break detected

*Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 7 **TXFNF**: TXFIFO not full
TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

An interrupt is generated if the TXFNFIE bit =1 in the USART_CR1 register.
0: Transmit FIFO is full
1: Transmit FIFO is not full

*Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

This bit is used during single buffer transmission.*

Bit 6 **TC**: Transmission complete
This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.
It is set by hardware when the transmission of a frame containing data is complete and when TXFE is set.
An interrupt is generated if TCIE=1 in the USART_CR1 register.
TC bit is cleared by software, by writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission is not complete
1: Transmission is complete

*Note: If TE bit is reset and no transmission is on going, the TC bit is immediately set.*
Bit 5 RXFNE: RXFIFO not empty
RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.
RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.
An interrupt is generated if RXFNEIE=1 in the USART_CR1 register.
0: Data is not received
1: Received data is ready to be read.

Bit 4 IDLE: Idle line detected
This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.
0: No Idle line is detected
1: Idle line is detected
Note: The IDLE bit is not set again until the RXFNE bit has been set (i.e. a new idle line occurs).
If Mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 ORE: Overrun error
This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.
An interrupt is generated if RXFNEIE=1 or EIE = 1 in the USART_CR1 register.
0: No overrun error
1: Overrun error is detected
Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.
This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.
Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the USART_ICR register.

0: No noise is detected
1: Noise is detected

**Note:** This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 53.5.8: Tolerance of the USART receiver to clock deviation on page 1970).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR1 register.

0: No Framing error is detected
1: Framing error or break character is detected

**Note:** This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error
1: Parity error

**Note:** This error is associated with the character in the USART_RDR.

### 53.7.10 USART interrupt and status register [alternate] (USART_ISR)

**Address offset:** 0x1C

**Reset value:** 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

**FIFO mode disabled**

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ABRF</td>
<td>ABRE</td>
<td>UDR</td>
<td>EOBF</td>
<td>RTOF</td>
<td>CTS</td>
<td>CTSIF</td>
<td>LBDI</td>
<td>TBER</td>
<td>TER</td>
<td>TXR</td>
<td>RXNE</td>
<td>IDLE</td>
<td>ORE</td>
<td>NE</td>
<td>FE</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

2028/2974 RM0455 Rev 3
Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT**: Transmission complete before guard time flag
This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.
It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE=1 in the USART_CR3 register.
This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)
1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

*Note:* If the USART does not support the Smartcard mode, this bit is reserved and kept at reset value. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is ‘1’. Refer to Section 53.4: USART implementation on page 1953.

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag
This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.
It can be used to verify that the USART is ready for reception before entering low-power mode.

*Note:* If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bit 21 **TEACK**: Transmit enable acknowledge flag
This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.
It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag
This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.
An interrupt is generated if WUFIE=1 in the USART_CR3 register.

*Note:* When UESM is cleared, WUF flag is also cleared.
The WUF interrupt is active only in low-power mode.
*If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 19 **RWU**: Receiver wakeup from Mute mode
This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.
When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.
0: Receiver in active mode
1: Receiver in Mute mode

*Note:* If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 1953.
Bit 18  **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

- 0: No break character transmitted
- 1: Break character transmitted

Bit 17  **CMF**: Character match flag

This bit is set by hardware, when a character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE=1 in the USART_CR1 register.

- 0: No Character match detected
- 1: Character Match detected

Bit 16  **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

- 0: USART is idle (no reception)
- 1: Reception on going

Bit 15  **ABRF**: Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 14  **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_CR3 register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 13  **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

- 0: No underrun error
- 1: underrun error

*Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 12  **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T=1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if the EOBIE=1 in the USART_CR2 register.

It is cleared by software, writing 1 to the EOBCF in the USART_ICR register.

- 0: End of Block not reached
- 1: End of Block (number of characters) reached

*Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*
Bit 11  **RTOF:** Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE=1 in the USART_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached
1: Timeout value reached without any data reception

**Note:** If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10  **CTS:** CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set
1: nCTS line reset

**Note:** If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9  **CTSF:** CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE=1 in the USART_CR3 register.

0: No change occurred on the nCTS status line
1: A change occurred on the nCTS status line

**Note:** If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8  **LBDF:** LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected
1: LIN break detected

**Note:** If the USART does not support LIN mode, this bit is reserved and kept at reset value.

Refer to Section 53.4: USART implementation on page 1953.

Bit 7  **TXE:** Transmit data register empty

TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in Smartcard T=0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register.

0: Data register full
1: Data register not full
Bit 6 TC: Transmission complete
   This bit indicates that the last data written in the USART_TDR has been transmitted out of
   the shift register.
   It is set by hardware when the transmission of a frame containing data is complete and
   when TXE is set.
   An interrupt is generated if TCIE=1 in the USART_CR1 register.
   TC bit is is cleared by software, by writing 1 to the TCCF in the USART_ICR register or by a
   write to the USART_TDR register.
   0: Transmission is not complete
   1: Transmission is complete
   Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.

Bit 5 RXNE: Read data register not empty
   RXNE bit is set by hardware when the content of the USART_RDR shift register has been
   transferred to the USART_RDR register. It is cleared by reading from the USART_RDR
   register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR
   register.
   An interrupt is generated if RXNEIE=1 in the USART_CR1 register.
   0: Data is not received
   1: Received data is ready to be read.

Bit 4 IDLE: Idle line detected
   This bit is set by hardware when an Idle Line is detected. An interrupt is generated if
   IDLEIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in
   the USART_ICR register.
   0: No Idle line is detected
   1: Idle line is detected
   Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line
   occurs).
   If Mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0),
   whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 ORE: Overrun error
   This bit is set by hardware when the data currently being received in the shift register is
   ready to be transferred into the USART_RDR register while RXNE=1. It is cleared by a
   software, writing 1 to the ORECF, in the USART_ICR register.
   An interrupt is generated if RXNEIE=1 or EIE = 1 in the USART_CR1 register.
   0: No overrun error
   1: Overrun error is detected
   Note: When this bit is set, the USART_RDR register content is not lost but the shift register is
   overwritten. An interrupt is generated if the ORE flag is set during multi buffer
   communication if the EIE bit is set.
   This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in
   the USART_CR3 register.
Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the USART_ICR register.

0: No noise is detected
1: Noise is detected

*Note:* This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

*When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 53.5.8: Tolerance of the USART receiver to clock deviation on page 1970).*

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR1 register.

0: No Framing error is detected
1: Framing error or break character is detected

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error
1: Parity error

### 53.7.11 USART interrupt flag clear register (USART_ICR)

**Address offset:** 0x20

**Reset value:** 0x0000 0000

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

Bit 20 **WUCF**: Wakeup from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

*Note:* If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART_ISR register.
Bits 16:14  Reserved, must be kept at reset value.

Bit 13  **UDRCF**: SPI slave underrun clear flag
Writing 1 to this bit clears the UDRF flag in the USART_ISR register.

*Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 12  **EOBCF**: End of block clear flag
Writing 1 to this bit clears the EOBF flag in the USART_ISR register.

*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 11  **RTOCF**: Receiver timeout clear flag
Writing 1 to this bit clears the RTOF flag in the USART_ISR register.

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 10  Reserved, must be kept at reset value.

Bit 9  **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.

*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 8  **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.

*Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 1953.*

Bit 7  **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6  **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5  **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4  **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3  **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2  **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1  **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0  **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.
53.7.12 USART receive data register (USART_RDR)

Address offset: 0x24
Reset value: 0x0000 0000

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see Figure 536).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

53.7.13 USART transmit data register (USART_TDR)

Address offset: 0x28
Reset value: 0x0000 0000

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value
Contains the data character to be transmitted.
The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 536).
When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

*Note: This register must be written only when TXE/TXFNF=1.*
53.7.14  USART prescaler register (USART_PRES)

This register can only be written when the USART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000 0000

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

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

**PRESCALER[3:0]**

- **Bits 31:4** Reserved, must be kept at reset value.
- **Bits 3:0** **PRESCALER[3:0]: Clock prescaler**
  - The USART input clock can be divided by a prescaler factor:
    - 0000: input clock not divided
    - 0001: input clock divided by 2
    - 0010: input clock divided by 4
    - 0011: input clock divided by 6
    - 0100: input clock divided by 8
    - 0101: input clock divided by 10
    - 0110: input clock divided by 12
    - 0111: input clock divided by 16
    - 1000: input clock divided by 32
    - 1001: input clock divided by 64
    - 1010: input clock divided by 128
    - 1011: input clock divided by 256
    - Remaining combinations: Reserved

**Note:** When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is 1011 i.e. input clock divided by 256.
### 53.7.15 USART register map

The table below gives the USART register map and reset values.

**Table 386. USART register map and reset values**

| Offset | Register name                                      | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 | 0x08 | 0x09 | 0x0A | 0x0B | 0x0C | 0x0D | 0x0E | 0x0F | 0x10 | 0x11 | 0x12 | 0x13 | 0x14 | 0x15 | 0x16 | 0x17 | 0x18 | 0x19 | 0x1A | 0x1B | 0x1C | 0x1D | 0x1E | 0x1F |
|--------|--------------------------------------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
|        | USART_CR1 FIFO enabled                           |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|        | USART_CR1 FIFO disabled                          |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|        | USART_CR2 ADD[7:0]                               |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|        | USART_CR3 TXFTCFG[2:0]                           |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|        | USART_BRR BRR[15:0]                              |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|        | USART_GTPR GT[7:0]                               |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|        | USART_RQR RXFRQ                                  |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|        | USART_ISR FIFO mode enabled                       |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|        | USART_ISR FIFO mode disabled                      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|        | USART_ICR                                       |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |

**Reset values:**

- 0x00: USART_CR1 FIFO enabled
- 0x04: USART_CR2 ADD[7:0]
- 0x08: USART_CR3 TXFTCFG[2:0]
- 0x0C: USART_BRR BRR[15:0]
- 0x10: USART_GTPR GT[7:0]
- 0x14: USART_RTOR RTO[23:0]
- 0x18: USART_RQR RXFRQ
- 0x1C: USART_ISR FIFO mode enabled
- 0x20: USART_ICR
Refer to Section 2.3: Memory organization for the register boundary addresses.

Table 386. USART register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|        | Reset value   | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
|        | Reset value   | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
|        | Reset value   | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
54 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitter (LPUART).

54.1 LPUART introduction

The LPUART is an UART which enables bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to enable UART communications up to 9600 baud/s. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports Half-duplex Single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.
54.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 baud/s to 9600 baud/s using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data
  Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wakeup from Mute mode by idle line detection or address mark detection

54.3 LPUART implementation

The following table shows LPUART implementation in comparison with USART.
Table 387. USART/LPUART features

<table>
<thead>
<tr>
<th>USART modes/features(^{(1)})</th>
<th>USART1/2/3/6/10</th>
<th>UART4/5/7/8/9</th>
<th>LPUART1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware flow control for modem</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Continuous communication using DMA</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Multiprocessor communication</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Synchronous mode (Master/Slave)</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Smartcard mode</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Single-wire Half-duplex communication</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IrDA SIR ENDEC block</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>LIN mode</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Dual clock domain and wakeup from low-power mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Receiver timeout interrupt</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Modbus communication</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Auto baud rate detection</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Driver Enable</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>USART data length</td>
<td></td>
<td>7, 8 and 9 bits</td>
<td></td>
</tr>
<tr>
<td>Tx/Rx FIFO</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Tx/Rx FIFO size</td>
<td></td>
<td></td>
<td>16</td>
</tr>
</tbody>
</table>

1. X = supported.
54.4 LPUART functional description

54.4.1 LPUART block diagram

The simplified block diagram given in Figure 563 shows two fully independent clock domains:

- **The lpuart_pclk clock domain**
  The lpuart_pclk clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.

- **The lpuart_ker_ck kernel clock domain**
  The lpuart_ker_ck is the LPUART clock source. It is independent of the lpuart_pclk and delivered by the RCC. So, the LPUART registers can be written/read even when the lpuart_ker_ck is stopped.
  
  When the dual clock domain feature is disabled, the lpuart_ker_ck is the same as the lpuart_pclk clock.

There is no constraint between lpuart_pclk and lpuart_ker_ck: lpuart_ker_ck can be faster or slower than lpuart_pclk, with no more limitation than the ability for the software to manage the communication fast enough.
54.4.2 LPUART signals

LPUART bidirectional communications requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)
  RX is the serial data input.

- **TX** (Transmit Data Output)
  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In Single-wire mode, this I/O is used to transmit and receive the data.

RS232 hardware flow control mode

The following pins are required in RS232 Hardware flow control mode:

- **CTS** (Clear To Send)
  When driven high, this signal blocks the data transmission at the end of the current transfer.

- **RTS** (Request to send)
  When it is low, this signal indicates that the USART is ready to receive data.

RS485 hardware flow control mode

The following pin is required in RS485 Hardware control mode:

- **DE** (Driver Enable)
  This signal activates the transmission mode of the external transceiver.

*Note:*  *DE and RTS share the same pin.*

54.4.3 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see Figure 537).

- 7-bit character length: M[1:0] = ‘10’
- 8-bit character length: M[1:0] = ‘00’
- 9-bit character length: M[1:0] = ‘01’

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

An *Idle character* is interpreted as an entire frame of “1”s. (The number of “1” ’s includes the number of stop bits).

A *Break character* is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.
54.4.4 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bits wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store
the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

**Note:** The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and Rx interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in LPUART_CR3 control register.

In this case:

- The RXFT flag is set in the LPUART_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bitfields.
  
  This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.

  RXFTCFG data have been received: one data in LPUART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to ‘101’, the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size - 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data does not set the overrun flag.

- The TXFT flag is set in the LPUART_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bitfields.
  
  This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

### 54.4.5 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

**Character transmission**

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see Figure 563).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

**Note:** The TE bit must be set before writing the data to be transmitted to the LPUART_TDR. The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame is sent after the TE bit is enabled.
Configurable stop bits

The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13,12).

- **1 stop bit**: This is the default value of number of stop bits.
- **2 Stop bits**: This is supported by normal LPUART, Single-wire and Modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = ‘00’) or 11 low bits (when M[1:0] = ‘01’) or 9 low bits (when M[1:0] = ‘10’) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

**Figure 565. Configurable stop bits**

Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to ‘1’.
5. Select DMA enable (DMAT) in LPUART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in Section 53.5.10: USART multiprocessor communication.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
   - When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   - When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART_TDR register, wait until TC=1. This indicates that the transmission of the last frame is complete.
   - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
- When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.
This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

**Single byte communication**

- **When FIFO mode disabled:**
  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:
  - the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;
  - the LPUART_TDR register is empty;
  - the next data can be written to the LPUART_TDR register without overwriting the previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- **When FIFO mode is enabled,** the TXFNF (TXFIFO Not Full) flag is set by hardware to indicate that:
  - the TXFIFO is not full;
  - the LPUART_TDR register is empty;
  - the next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

  When the TXFIFO is not full, the TXFNF flag stays at '1' even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if TXFNEIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE is case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

  After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC=1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see Figure 566: TC/TXE behavior when transmitting).
Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see Figure 564). If a ‘1’ is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

54.4.6 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still ‘0’. If the start sample is at ‘1’, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.
Character reception

During an LPUART reception, data are shifted in least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART.CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to ‘1’.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in Section 53.5.10: USART multiprocessor communication.
6. Set the RE bit LPUART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the Receive Data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty i.e. there is a data in the RXFIFO to be read.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set.
Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

**Break character**

When a break character is received, the USART handles it as a framing error.

**Idle character**

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

**Overrun error**

- **FIFO mode disabled**
  
  An overrun error occurs when a character is received when RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.
  
  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
  
  - the ORE bit is set;
  - the RDR content is not lost. The previous data is available when a read to LPUART_RDR is performed;
  - the shift register is overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXNEIE bit or EIE bit is set.

- **FIFO mode enabled**
  
  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.
  
  Data can not be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.
  
  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:
  
  - the ORE bit is set;
  - the first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  - the shift register is overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECf bit in the ICR register.

**Note:**

The ORE bit, when set, indicates that at least 1 data has been lost. 

When the FIFO mode is disabled, there are two possibilities

- if RXNE=1, then the last valid data is stored in the receive register (RDR) and can be read,

- if RXNE=0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.
Selecting the clock source

The choice of the clock source is done through the Clock Control system (see Section Reset and clock controller (RCC)). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wakeup from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see Section Reset and clock controller (RCC)). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRESC register.

Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming baud as close as possible to the middle of the baud-period. Only a single sample is taken of each of the incoming bauds.

Note: There is no noise detection for data.

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the LPUART_RDR register.
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of
multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

**Configurable stop bits during reception**

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample i.e. during the second stop bit. The first stop bit is not checked for framing error.

### 54.4.7 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

\[
\text{Tx/Rx baud} = \frac{256 \times \text{lpuartckpres}}{\text{LPUARTDIV}}
\]

LPUARTDIV is defined in the LPUART_BRR register.

**Note:** The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence the baud rate register value should not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

\( f_{CK} \) must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 baud. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock. For example, if the LPUART clock source frequency is 100 MHz, the maximum baud rate that can be reached is about 33 Mbaud.

**Table 388. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32,768 KHz**

<table>
<thead>
<tr>
<th>S.No</th>
<th>Desired</th>
<th>Actual</th>
<th>Value programmed in the baud rate register</th>
<th>% Error (Calculated - Desired) B.rate / Desired B.rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3 KBps</td>
<td>0.3 KBps</td>
<td>0x6D3A</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.6 KBps</td>
<td>0.6 KBps</td>
<td>0x369D</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1200 Bps</td>
<td>1200.087 Bps</td>
<td>0x1B4E</td>
<td>0.007</td>
</tr>
<tr>
<td>4</td>
<td>2400 Bps</td>
<td>2400.17 Bps</td>
<td>0xDA7</td>
<td>0.007</td>
</tr>
<tr>
<td>5</td>
<td>4800 Bps</td>
<td>4801.72 Bps</td>
<td>0x6D3</td>
<td>0.035</td>
</tr>
<tr>
<td>6</td>
<td>9600 KBps</td>
<td>9608.94 Bps</td>
<td>0x369</td>
<td>0.093</td>
</tr>
</tbody>
</table>
54.4.8 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes which contribute to the total deviation are:

- **DTRA**: deviation due to the transmitter error (which also includes the deviation of the transmitter’s local oscillator)
- **DQUANT**: error due to the baud rate quantization of the receiver
- **DREC**: deviation of the receiver local oscillator
- **DTCL**: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

\[
DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}
\]

where

- **DWU** is the error due to sampling point deviation when the wakeup from low-power mode is used.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in Table 390:

- Number of Stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

### Table 389. Error calculation for programmed baud rates at f\(_\text{CK}\) = 100 MHz

<table>
<thead>
<tr>
<th>S.No</th>
<th>Desired</th>
<th>Actual</th>
<th>Value programmed in the baud rate register</th>
<th>% Error = (Calculated - Desired) B.rates / Desired B.rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>38400 Baud</td>
<td>38400,04 Baud</td>
<td>A2C2A</td>
<td>0,0001</td>
</tr>
<tr>
<td>2</td>
<td>57600 Baud</td>
<td>57600,06 Baud</td>
<td>6C81C</td>
<td>0,0001</td>
</tr>
<tr>
<td>3</td>
<td>115200 Baud</td>
<td>115200,12 Baud</td>
<td>3640E</td>
<td>0,0001</td>
</tr>
<tr>
<td>4</td>
<td>230400 Baud</td>
<td>230400,23 Baud</td>
<td>1B207</td>
<td>0,0001</td>
</tr>
<tr>
<td>5</td>
<td>460800 Baud</td>
<td>460804,61 Baud</td>
<td>D903</td>
<td>0,001</td>
</tr>
<tr>
<td>6</td>
<td>921600 Baud</td>
<td>921625,81 Baud</td>
<td>6C81</td>
<td>0,0028</td>
</tr>
<tr>
<td>7</td>
<td>4000 KBaud</td>
<td>4000000,00 Baud</td>
<td>1900</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>10000 Kbaud</td>
<td>10000000,00 Baud</td>
<td>A00</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>20000 Kbaud</td>
<td>20000000,00 Baud</td>
<td>500</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>30000 Kbaud</td>
<td>33032258,06 Baud</td>
<td>307</td>
<td>0,1</td>
</tr>
</tbody>
</table>
54.4.9 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all non addressed receivers.

The non addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_ker_ck cycles), otherwise Mute mode might remain active.

When the Mute mode is enabled:
- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in LPUART_ISR register is set to ‘1’. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The LPUART enters Mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.
The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of Mute mode behavior using Idle line detection is given in Figure 568.

**Figure 568. Mute mode using Idle line detection**

![Figure 568](MSv31154V1)

**Note:**
- If the MMRQ is set while the IDLE character has already elapsed, the Mute mode is not entered (RWU is not set).
- If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

### 4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’ otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the LPUART_CR2 register.

**Note:**
In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters Mute mode.

The LPUART also enters Mute mode when the MMRQ bit is written to ‘1’. The RWU bit is also automatically set in this case.

The LPUART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

**Note:**
When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in Mute mode.

An example of Mute mode behavior using address mark detection is given in Figure 569.
54.4.10 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in Table 391.

### Table 391: LPUART frame formats

<table>
<thead>
<tr>
<th>M bits</th>
<th>PCE bit</th>
<th>LPUART frame(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

### Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

### Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).
Parity checking in reception
If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission
If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

54.4.11 LPUART single-wire Half-duplex communication
Single-wire Half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register. In this mode, the following bits must be kept cleared:
- LINEN and CLKEN bits in the LPUART_CR2 register,
- SCEN and IREN bits in the LPUART_CR3 register.

The LPUART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to ‘1’:
- The TX and RX lines are internally connected.
- The TX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

54.4.12 Continuous communication using DMA and LPUART
The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 53.4: USART implementation on page 1953 to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 53.5.6. To perform continuous communication. When FIFO is disabled, you can clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA
DMA mode can be enabled for transmission by setting DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding Direct memory access controller section) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.

2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.

3. Configure the total number of bytes to be transferred to the DMA control register.

4. Configure the channel priority in the DMA register.

5. Configure DMA interrupt generation after half/full transfer as required by the application.

6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.

7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the LPUART communication is complete. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

**Figure 570. Transmission using DMA**

**Note:** When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in LPUART.CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to the corresponding Direct memory access controller (DMA) section) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register.
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 571. Reception using DMA

Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt.
enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

54.4.13 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The Figure 558 shows how to connect 2 devices in this mode:

RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. Figure 573 shows an example of communication with RTS flow control enabled.

Note: When FIFO mode is enabled, nRTS is deasserted only when RXFIFO is full.
RS232 CTS flow control

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. Figure 574 shows an example of communication with CTS flow control enabled.

**Figure 574. RS232 CTS flow control**

- **nCTS**
- **Transmit data register**
- **TDR**
  - Data 2
  - empty
  - Data 3
  - empty
- **TX**
  - Data 1
  - Stop bit
  - Start bit
  - Data 2
  - Stop bit
  - idle
  - Start bit
  - Data 3
- **Writing data 3 in TDR**
- **Transmission of Data 3 is delayed until nCTS = 0**

**Note:** For correct behavior, nCTS must be asserted at least 3 LPUART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.
The LPUART DEAT and DEDT are expressed in LPUART clock source (fCK) cycles:

- The Driver enable assertion time equals
  - \((1 + (\text{DEAT} \times P)) \times f_{\text{CK}}, \text{if } P \neq 0\)
  - \((1 + \text{DEAT}) \times f_{\text{CK}}, \text{if } P = 0\)
- The Driver enable deassertion time equals
  - \((1 + (\text{DEDT} \times P)) \times f_{\text{CK}}, \text{if } P \neq 0\)
  - \((1 + \text{DEDT}) \times f_{\text{CK}}, \text{if } P = 0\)

where \(P = \text{BRR}[20:11]\)

### 54.4.14 LPUART low-power management

The LPUART has advanced low-power mode functions that enable it to transfer properly data even when the lpuart_pclk clock is disabled. The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set. When the usart_pclk is gated, the LPUART provides a wakeup interrupt (\text{usart_wkup}) if a specific action requiring the activation of the usart_pclk clock is needed:

- If FIFO mode is disabled
  lpuart_pclk clock has to be activated to empty the LPUART data register.
  In this case, the lpuart_wkup interrupt source is the RXNE set to ‘1’. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled
  lpuart_pclk clock has to be activated
  - to fill the TXFIFO
  - or to empty the RXFIFO
  In this case, the lpuart_wkup interrupt source can be:
    - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
    - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
    - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

To avoid overrun/underrun errors and transmit/receive data in low-power mode, the lpuart_wkup interrupt source can be one of the following events:
- TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
- RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

For example, the application can set the threshold to the maximum RXFIFO size if the wakeup time is less than the time to receive a single byte across the line.

Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wakeup the MCU from low-power mode enables doing as many LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.
Alternatively, a specific `lpuart_wkup` interrupt may be selected through the WUS bitfields. When the wakeup event is detected, the WUF flag is set by hardware and `lpuart_wkup` interrupt is generated if the WUFIE bit is set. In this case the `lpuart_wkup` interrupt is not mandatory for the wakeup. The WUF being set is sufficient to wakeup the MCU from low-power mode.

**Note:**
Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in low-power or in an active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the LPUART is actually enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled upon exit from low-power mode.

When FIFO is enabled, the wakeup from low-power mode on address match is only possible when Mute mode is enabled.

### Using Mute mode with low-power mode

If the LPUART is put into Mute mode before entering low-power mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wakeup from Mute mode on address match is used, then the low-power mode wakeup source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in Mute mode upon address match and wake up from low-power mode.

**Note:**
When FIFO management is enabled, Mute mode is used with wakeup from low-power mode without any constraints (i.e. the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

### Wakeup from low-power mode when LPUART kernel clock lpuart_ker_ck is OFF in low-power mode

If during low-power mode, the lpuart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the lpuart_ker_ck clock to be switched ON thanks to the lpuart_ker_ck_req signal. The lpuart_ker_ck is then used for the frame reception.

If the wakeup event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wakeup event is not verified, the usart_ker_ck is switched OFF again, the MCU is not waken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of wakeup event programmed to “address match detection” and FIFO management disabled.

*Figure 575* shows the behavior when the wakeup event is verified.
Figure 575. Wakeup event verified (wakeup event = address match, FIFO disabled)

Address match event WUF = '1'
LPUART sends a wakeup event to the MCU

Data reception goes on

Figure 576 shows the behavior when the wakeup event is not verified.

Figure 576. Wakeup event not verified (wakeup event = address match, FIFO disabled)

Address does not match

Note: The above figures are valid when address match or any received frame is used as wakeup event. In the case the wakeup event is the start bit detection, the LPUART sends the wakeup event to the MCU at the end of the start bit.
Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wakeup time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see Section 54.4.8: Tolerance of the LPUART receiver to clock deviation).

Let us take the example of OVER8 = 0, M bits = '01', ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to Table 390: Tolerance of the LPUART receiver, the LPUART receiver tolerance equals 3.41%.

\[
D_{WU} = t_{WU\text{LPUART}} / (11 \times T_{bit \text{ Min}})
\]

T_{bit \text{ Min}} = t_{WU\text{LPUART}} / (11 \times D_{WU})

where \( t_{WU\text{LPUART}} \) is the wakeup time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

\[
t_{WU\text{LPUART}} = 3 \, \mu s \text{ (values provided only as examples; for correct values, refer to the device datasheet).}
\]

\[
D_{WU\max} = 3.41\% - 1\% = 2.41\%
\]

\[
T_{bit \min} = 3 \, \mu s / (11 \times 2.41\%) = 11.32 \, \mu s.
\]

As a result, the maximum baud rate that enables to wakeup correctly from low-power mode is: \( 1/11.32 \, \mu s = 88.36 \, \text{Kbaud} \).

54.5 LPUART interrupts

Refer to Table 385 for a detailed description of all LPUART interrupt requests.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Interrupt activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit data register empty</td>
<td>TXE</td>
<td>TXEIE</td>
<td>TXE cleared when a data is written in TDR</td>
<td>YES</td>
</tr>
<tr>
<td>Transmit FIFO Not Full</td>
<td>TXFNF</td>
<td>TXFNFIE</td>
<td>TXFNF cleared when TXFIFO is full.</td>
<td>YES</td>
</tr>
<tr>
<td>Transmit FIFO Empty</td>
<td>TXFE</td>
<td>TXFEIE</td>
<td>TXFE cleared when the TXFIFO contains at least one data or by setting TXFRQ bit.</td>
<td>YES</td>
</tr>
<tr>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFTIE</td>
<td>TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold</td>
<td>YES</td>
</tr>
</tbody>
</table>
### Table 392. LPUART interrupt requests (continued)

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Interrupt activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTS interrupt</td>
<td>CTSIF</td>
<td>CTSIE</td>
<td>CTSIF cleared by software by setting CTSCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Transmission Complete</td>
<td>TC</td>
<td>TCIE</td>
<td>TC cleared when a data is written in TDR or by setting TCCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Receive data register not empty (data ready to be read)</td>
<td>RXNE</td>
<td>RXNEIE</td>
<td>RXNE cleared by reading RDR or by setting RXFRQ bit.</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO Not Empty</td>
<td>RXFNE</td>
<td>RXFNEIE</td>
<td>RXFNE cleared when the RXFIFO is empty or by setting RXFRQ bit.</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO Full</td>
<td>RXFF(1)</td>
<td>RXFFIE</td>
<td>RXFF cleared when the RXFIFO contains at least one data.</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFTIE</td>
<td>RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold</td>
<td>YES YES</td>
</tr>
<tr>
<td>Overrun error detected</td>
<td>ORE</td>
<td>RX-NEIE/RXFNEIE</td>
<td>ORE cleared by setting ORECF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Idle line detected</td>
<td>IDLE</td>
<td>IDLEIE</td>
<td>IDLE cleared by setting IDLECF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Parity error</td>
<td>PE</td>
<td>PEIE</td>
<td>PE cleared by setting PECF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Noise Flag, Overrun error and Framing Error in multibuffer communication.</td>
<td>NE or ORE or FE</td>
<td>EIE</td>
<td>NE cleared by setting NECF bit. ORE cleared by setting ORECF bit. FE flag cleared by setting FECF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Character match</td>
<td>CMF</td>
<td>CMIE</td>
<td>CMF cleared by setting CMCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Wakeup from low-power mode</td>
<td>WUF(2)</td>
<td>WUFIE</td>
<td>WUF is cleared by setting WUCF bit.</td>
<td>NO YES</td>
</tr>
<tr>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFTIE</td>
<td>TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFTIE</td>
<td>RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold.</td>
<td>YES YES</td>
</tr>
</tbody>
</table>

1. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register is not written, and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
2. The WUF interrupt is active only in low-power mode.

54.6 LPUART registers

Refer to Section 1.2 on page 97 for a list of abbreviations used in register descriptions. The peripheral registers have to be accessed by words (32 bits).

54.6.1 LPUART control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00
Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>RXFFIE: RXFIFO Full interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt is inhibited</td>
</tr>
<tr>
<td></td>
<td>1: An LPUART interrupt is generated when RXFF=1 in the LPUART_ISR register</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>TXFEIE: TXFIFO empty interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt is inhibited</td>
</tr>
<tr>
<td></td>
<td>1: An LPUART interrupt is generated when TXFE=1 in the LPUART_ISR register</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>FIFOEN: FIFO mode enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: FIFO mode is disabled.</td>
</tr>
<tr>
<td></td>
<td>1: FIFO mode is enabled.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>M1: Word length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.</td>
</tr>
<tr>
<td></td>
<td>M[1:0] = '00': 1 Start bit, 8 Data bits, n Stop bit</td>
</tr>
<tr>
<td></td>
<td>M[1:0] = '01': 1 Start bit, 9 Data bits, n Stop bit</td>
</tr>
<tr>
<td></td>
<td>M[1:0] = '10': 1 Start bit, 7 Data bits, n Stop bit</td>
</tr>
<tr>
<td></td>
<td>This bit can only be written when the LPUART is disabled (UE=0).</td>
</tr>
<tr>
<td></td>
<td>Note: In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31:30</th>
<th>M1</th>
<th>FIFOEN</th>
<th>DEAT[4:0]</th>
<th>DEET[4:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 27:26: Reserved, must be kept at reset value.
Bits 25:21 **DEAT[4:0]:** Driver Enable assertion time
This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer [Section 53.5.20: RS232 Hardware flow control and RS485 Driver Enable](#). This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 20:16 **DEDT[4:0]:** Driver Enable deassertion time
This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in lpuart_ker_ck clock cycles. For more details, refer [Section 54.4.13: RS232 Hardware flow control and RS485 Driver Enable](#).

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE:** Character match interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME:** Mute mode enable
This bit activates the Mute mode function of the LPUART. When set, the LPUART can switch between the active and Mute modes, as defined by the WAKE bit. It is set and cleared by software.
0: Receiver in active mode permanently
1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0:** Word length
This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE=0).

Bit 11 **WAKE:** Receiver wakeup method
This bit determines the LPUART wakeup method from Mute mode. It is set or cleared by software.
0: Idle line
1: Address mark
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 10 **PCE:** Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software.
Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 9 **PS:** Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the LPUART is disabled (UE=0).
Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE=1 in the LPUART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE/TXFNF =1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC=1 in the LPUART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE=1 or RXNE/RXFNE=1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE=1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

**Note:** During transmission, a low pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

When TE is set there is a 1 bit-time delay before the transmission starts.
54.6.2 LPUART control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00
Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

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

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.
Bits 31:30  Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

\[ M[1:0] = '00': \text{1 Start bit, 8 Data bits, n Stop bit} \]
\[ M[1:0] = '01': \text{1 Start bit, 9 Data bits, n Stop bit} \]
\[ M[1:0] = '10': \text{1 Start bit, 7 Data bits, n Stop bit} \]
This bit can only be written when the LPUART is disabled (UE=0).

*Note: In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.*

Bits 27:26  Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time
This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer Section 53.5.20: RS232 Hardware flow control and RS485 Driver Enable. This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time
This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in lpuart_ker_ck clock cycles. For more details, refer Section 54.4.13: RS232 Hardware flow control and RS485 Driver Enable. If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed. This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15  Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable
This bit activates the Mute mode function of the LPUART. When set, the LPUART can switch between the active and Mute modes, as defined by the WAKE bit. It is set and cleared by software.
0: Receiver in active mode permanently
1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length
This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description). This bit can only be written when the LPUART is disabled (UE=0).
Bit 11 **WAKE**: Receiver wakeup method

This bit determines the LPUART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line
1: Address mark

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled
1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity
1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever PE=1 in the LPUART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited
1: A LPUART interrupt is generated whenever TXE/TXFNF =1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever TC=1 in the LPUART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

0: Interrupt is inhibited
1: A LPUART interrupt is generated whenever ORE=1 or RXNE/RXFNE=1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever IDLE=1 in the LPUART_ISR register
Bit 3 TE: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

Note: During transmission, a low pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.
When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 RE: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 UESM: LPUART enable in Stop mode
When this bit is cleared, the LPUART is not able to wake up the MCU from low-power mode.
When this bit is set, the LPUART is able to wake up the MCU from low-power mode, provided that the LPUART clock selection is HSI or LSE in the RCC.
This bit is set and cleared by software.
0: LPUART not able to wake up the MCU from low-power mode.
1: LPUART able to wake up the MCU from low-power mode. When this function is active, the clock source for the LPUART must be HSI or LSE (see RCC chapter)

Note: It is recommended to set the UESM bit just before entering low-power mode and clear it on exit from low-power mode.

Bit 0 UE: LPUART enable
When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.
0: LPUART prescaler and outputs disabled, low-power mode
1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

54.6.3 LPUART control register 2 (LPUART_CR2)
Address offset: 0x04
Reset value: 0x0000 0000
Bits 31:24  **ADD[7:0]**: Address of the LPUART node

**ADD[7:4]**:
- These bits give the address of the LPUART node or a character code to be recognized.
- They are used to wake up the MCU with 7-bit address mark detection in multiprocessor communication during Mute mode or Stop mode. The MSB of the character sent by the transmitter should be equal to 1. They can also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.
- These bits can only be written when reception is disabled (RE = 0) or the LPUART is disabled (UE=0)

**ADD[3:0]**:
- These bits give the address of the LPUART node or a character code to be recognized.
- They are used for wakeup with address mark detection in multiprocessor communication during Mute mode or low-power mode.
- These bits can only be written when reception is disabled (RE = 0) or the LPUART is disabled (UE=0)

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

Bit 19  **MSBFIRST**: Most significant bit first

- This bit is set and cleared by software.
- 0: data is transmitted/received with data bit 0 first, following the start bit.
- 1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
- This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 18  **DATAINV**: Binary data inversion

- This bit is set and cleared by software.
- 0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
- 1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
- This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 17  **TXINV**: TX pin active level inversion

- This bit is set and cleared by software.
- 0: TX pin signal works using the standard logic levels (VDD =1/idle, Gnd=0/mark)
- 1: TX pin signal values are inverted. ((VDD =0/mark, Gnd=1/idle).
- This enables the use of an external inverter on the TX line.
- This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 16  **RXINV**: RX pin active level inversion

- This bit is set and cleared by software.
- 0: RX pin signal works using the standard logic levels (VDD =1/idle, Gnd=0/mark)
- 1: RX pin signal values are inverted. ((VDD =0/mark, Gnd=1/idle).
- This enables the use of an external inverter on the RX line.
- This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15  **SWAP**: Swap TX/RX pins

- This bit is set and cleared by software.
- 0: TX/RX pins are used as defined in standard pinout
- 1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
- This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 14  Reserved, must be kept at reset value.
RM0455 Low-power universal asynchronous receiver transmitter (LPUART)

Bits 13:12  **STOP[1:0]**: STOP bits
   These bits are used for programming the stop bits.
   00: 1 stop bit
   01: Reserved.
   10: 2 stop bits
   11: Reserved
   This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 11:5  Reserved, must be kept at reset value.

Bit 4  **ADDM7**:7-bit Address Detection/4-bit Address Detection
   This bit is for selection between 4-bit address detection or 7-bit address detection.
   0: 4-bit address detection
   1: 7-bit address detection (in 8-bit data mode)
   This bit can only be written when the LPUART is disabled (UE=0)
   **Note:** In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0  Reserved, must be kept at reset value.

### 54.6.4  LPUART control register 3 (LPUART_CR3)

Address offset: 0x08
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DEP</td>
<td>DEM</td>
<td>DDRE</td>
<td>OVRDI</td>
<td>Res.</td>
<td>CTSIE</td>
<td>CTSE</td>
<td>RTSE</td>
<td>DMAT</td>
<td>DMAR</td>
<td>Res.</td>
<td>Res.</td>
<td>HDSEL</td>
<td>Res.</td>
<td>Res.</td>
<td>EIE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:29 **TXFTCFG[2:0]**: TXFIFO threshold configuration

- 000: TXFIFO reaches 1/8 of its depth.
- 001: TXFIFO reaches 1/4 of its depth.
- 110: TXFIFO reaches 1/2 of its depth.
- 100: TXFIFO reaches 7/8 of its depth.
- 101: TXFIFO becomes empty.
- Remaining combinations: Reserved.

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]**: Receive FIFO threshold configuration

- 000: Receive FIFO reaches 1/8 of its depth.
- 001: Receive FIFO reaches 1/4 of its depth.
- 110: Receive FIFO reaches 1/2 of its depth.
- 011: Receive FIFO reaches 3/4 of its depth.
- 100: Receive FIFO reaches 7/8 of its depth.
- 101: Receive FIFO becomes full.
- Remaining combinations: Reserved.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wakeup from low-power mode interrupt enable
This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever WUF=1 in the LPUART_ISR register

**Note:** **WUFIE must be set before entering in low-power mode.**

The WUF interrupt is active only in low-power mode.

**Note:** If the LPUART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to **Section 53.4: USART implementation.**

Bits 21:20 **WUS[1:0]**: Wakeup from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (Wakeup from low-power mode flag).

- 00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
- 01: Reserved.
- 10: WUF active on Start bit detection
- 11: WUF active on RXNE.

This bitfield can only be written when the LPUART is disabled (UE=0).

**Note:** If the LPUART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to **Section 53.4: USART implementation.**

Bits 19:16 Reserved, must be kept at reset value.
Bit 15 **DEP**: Driver enable polarity selection
- 0: DE signal is active high.
- 1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
- 0: DE function is disabled.
- 1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 13 **DDRE**: DMA Disable on Reception Error
- 0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
- 1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE=0).
*Note: The reception errors are: parity error, framing error or noise error.*

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
- 0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
- 1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE=0).
*Note: This control bit enables checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSE**: CTS interrupt enable
- 0: Interrupt is inhibited
- 1: An interrupt is generated whenever CTSIF=1 in the LPUART_ISR register

Bit 9 **RTSE**: CTS enable
- 0: CTS hardware flow control disabled
- 1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.
This bit can only be written when the LPUART is disabled (UE=0)

Bit 8 **RTSE**: RTS enable
- 0: RTS hardware flow control disabled
- 1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
- 1: DMA mode is enabled for transmission
- 0: DMA mode is disabled for transmission
54.6.5 LPUART baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:20</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 19:0</td>
<td><strong>BRR[19:0]</strong>: LPUART baud rate</td>
</tr>
</tbody>
</table>

*Note:* It is forbidden to write values lower than 0x300 in the LPUART_BRR register. Provided that LPUART_BRR must be ≥ 0x300 and LPUART_BRR is 20 bits, a care should be taken when generating high baud rates using high fck values. fck must be in the range [3 x baud rate..4096 x baud rate].
### 54.6.6 LPUART request register (LPUART_RQR)

Address offset: 0x18  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Default Value</th>
<th>Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:5</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>TXFRQ: Transmit data flush request</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>RXFRQ: Receive data flush request</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the RXNE flag. This enables discarding the received data without reading it, and avoid an overrun condition.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>MMRQ: Mute mode request</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SBKRQ: Send break request</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 54.6.7 LPUART interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C  
Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

#### FIFO mode enabled

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Default Value</th>
<th>Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:5</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>TXFT</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>RXFT</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>RXFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>TXFE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>REACK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>TEACK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>WUF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>RUW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>SBKF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CMF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>BUSY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 27 **TXFT**: TXFIFO threshold flag
This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG in LPUART_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the LPUART_CR3 register.

0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag
This bit is set by hardware when the RXFIFO reaches the threshold programmed in RXFTCFG in LPUART_CR3 register i.e. the Receive FIFO contains RXFTCFG data. An interrupt is generated if the RXFTIE bit = 1 (bit 27) in the LPUART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO full
This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register. An interrupt is generated if the RXFFIE bit = 1 in the LPUART_CR1 register.

0: RXFIFO is not full
1: RXFIFO is full

Bit 23 **TXFE**: TXFIFO empty
This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit = 1 in the LPUART_CR1 register.

0: TXFIFO is not empty
1: TXFIFO is empty

Bit 22 **REACK**: Receive enable acknowledge flag
This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

**Note**: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag
This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the LPUART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag
This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.

An interrupt is generated if WUFIE=1 in the LPUART_CR3 register.

**Note**: When UESM is cleared, WUF flag is also cleared.

The WUF interrupt is active only in low-power mode.

If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.
Bit 19 **RWU**: Receiver wakeup from Mute mode

This bit indicates if the LPUART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in Active mode
1: Receiver in Mute mode

*Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.*

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRO bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted
1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE=1 in the LPUART_CR1 register.

0: No Character match detected
1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)
1: Reception on going

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

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set
1: nCTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE=1 in the LPUART_CR3 register.

0: No change occurred on the nCTS status line
1: A change occurred on the nCTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 Reserved, must be kept at reset value.
Bit 7 **TXFNF**: TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit = 1 in the LPUART_CR1 register.
0: Data register is full/Transmit FIFO is full.
1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXFF is set. An interrupt is generated if TCIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the TCCF in the LPUART_ICR register or by a write to the LPUART_TDR register.

An interrupt is generated if TCIE=1 in the LPUART_CR1 register.

0: Transmission is not complete
1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXFNEIE=1 in the LPUART_CR1 register.

0: Data is not received
1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected
1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (i.e. a new idle line occurs).

If Mute mode is enabled (MME=1), IDLE is set if the LPUART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.
Bit 3 **ORE**: Overrun error
This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.
An interrupt is generated if RXFNEIE=1 or EIE = 1 in the LPUART_CR1 register.
0: No overrun error
1: Overrun error is detected

*Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE**: Start bit noise detection flag
This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NECF bit in the LPUART_ICR register.
0: No noise is detected
1: Noise is detected

*Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.*

This error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error
This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.
When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the LPUART_CR1 register.
0: No Framing error is detected
1: Framing error or break character is detected

*Note: This error is associated with the character in the LPUART_RDR.*

Bit 0 **PE**: Parity error
This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.
An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.
0: No parity error
1: Parity error

*Note: This error is associated with the character in the LPUART_RDR.*
54.6.8  LPUART interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C
Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:23</td>
<td>Reserved, must be kept at reset value.</td>
<td>r</td>
</tr>
<tr>
<td>22</td>
<td>REACK: Receive enable acknowledge flag</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>This bit is set/reset by hardware when the Receive Enable value is taken into account by the LPUART.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>It can be used to verify that the LPUART is ready for reception before entering low-power mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>TEACK: Transmit enable acknowledge flag</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the LPUART_CR1 register, in order to respect the TE=0 minimum period.</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>WUF: Wakeup from low-power mode flag</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register. An interrupt is generated if WUFIE=1 in the LPUART_CR3 register.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: When USEM is cleared, WUF flag is also cleared.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The WUF interrupt is active only in low-power mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>RWU: Receiver wakeup from Mute mode</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>This bit indicates if the LPUART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register. When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Receiver in active mode</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>1: Receiver in Mute mode</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>
Bit 18  **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted
1: Break character transmitted

Bit 17  **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE=1 in the LPUART_CR1 register.

0: No Character match detected
1: Character Match detected

Bit 16  **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)
1: Reception on going

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

Bit 10  **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set
1: nCTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9  **CTSIF**: CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE=1 in the LPUART_CR3 register.

0: No change occurred on the nCTS status line
1: A change occurred on the nCTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8  Reserved, must be kept at reset value.

Bit 7  **TXE**: Transmit data register empty/TXFIFO not full

TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.

An interrupt is generated if the TXEIE bit =1 in the LPUART_CR1 register.

0: Data register full
1: Data register not full

*Note: This bit is used during single buffer transmission.*
Bit 6  TC: Transmission complete
This bit is set by hardware if the transmission of a frame containing data is complete and if
TXE is set. An interrupt is generated if TCIE=1 in the LPUART_CR1 register. It is cleared by
software, writing 1 to the TCCF in the LPUART_ICR register or by a write to the
LPUART_TDR register.
An interrupt is generated if TCIE=1 in the LPUART_CR1 register.
0: Transmission is not complete
1: Transmission is complete
Note: If TE bit is reset and no transmission is on going, the TC bit is immediately set.

Bit 5  RXNE: Read data register not empty
RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been
transferred to the LPUART_RDR register. It is cleared by reading from the LPUART_RDR
register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the
LPUART_RQR register.
An interrupt is generated if RXNEIE=1 in the LPUART_CR1 register.
0: Data is not received
1: Received data is ready to be read.

Bit 4  IDLE: Idle line detected
This bit is set by hardware when an Idle Line is detected. An interrupt is generated if
IDLEIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in
the LPUART_ICR register.
0: No Idle line is detected
1: Idle line is detected
Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line
occurs).
If Mute mode is enabled (MME=1), IDLE is set if the LPUART is not mute (RWU=0),
whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3  ORE: Overrun error
This bit is set by hardware when the data currently being received in the shift register is
ready to be transferred into the LPUART_RDR register while RXNE=1. It is cleared by a
software, writing 1 to the ORECF, in the LPUART_ICR register.
An interrupt is generated if RXNEIE=1 or EIE = 1 in the LPUART_CR1 register.
0: No overrun error
1: Overrun error is detected
Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register
is overwritten. An interrupt is generated if the ORE flag is set during multi buffer
communication if the EIE bit is set.
This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in
the LPUART_CR3 register.
Bit 2 **NE**: Start bit noise detection flag
This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NECF bit in the LPUART_ICR register.

0: No noise is detected
1: Noise is detected

*Note*: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

Bit 1 **FE**: Framing error
This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register. When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame). An interrupt is generated if EIE = 1 in the LPUART_CR1 register.

0: No Framing error is detected
1: Framing error or break character is detected

Bit 0 **PE**: Parity error
This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register. An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error
1: Parity error

### 54.6.9 LPUART interrupt flag clear register (LPUART_ICR)
Address offset: 0x20
Reset value: 0x0000 0000

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

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wakeup from low-power mode clear flag
Writing 1 to this bit clears the WUF flag in the LPUART_ISR register.

*Note*: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag
Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bits 8:7 Reserved, must be kept at reset value.
Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

### 54.6.10 LPUART receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see Figure 563).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

### 54.6.11 LPUART transmit data register (LPUART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

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

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

**rw** rw rw rw rw rw rw rw
Bits 31:9  Reserved, must be kept at reset value.

Bits 8:0  **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 563).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART.CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

*Note: This register must be written only when TXE/TXFNF=1.*

### 54.6.12  LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE=0).

**Address offset:** 0x2C

**Reset value:** 0x0000 0000

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

**PRESCALER[3:0]**:

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

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

**Bits 3:0  **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

- 0000: input clock not divided
- 0001: input clock divided by 2
- 0010: input clock divided by 4
- 0011: input clock divided by 6
- 0100: input clock divided by 8
- 0101: input clock divided by 10
- 0110: input clock divided by 12
- 0111: input clock divided by 16
- 1000: input clock divided by 32
- 1001: input clock divided by 64
- 1010: input clock divided by 128
- 1011: input clock divided by 256

Remaining combinations: Reserved.

*Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is 1011 i.e. input clock divided by 256.*
54.6.13  LPUART register map

The table below gives the LPUART register map and reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>LPUART_CR1</td>
<td>0x00</td>
<td>LPUART_CR1</td>
</tr>
<tr>
<td></td>
<td>FIFO mode enabled</td>
<td></td>
<td>FIFO mode enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_CR2</td>
<td>0x04</td>
<td>LPUART_CR2</td>
</tr>
<tr>
<td></td>
<td>ADD[7:0]</td>
<td></td>
<td>ADD[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_CR3</td>
<td>0x08</td>
<td>LPUART_CR3</td>
</tr>
<tr>
<td></td>
<td>TXFTCFG[2:0]</td>
<td></td>
<td>TXFTCFG[2:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_BRR</td>
<td>0x0C</td>
<td>LPUART_BRR</td>
</tr>
<tr>
<td></td>
<td>BRR[19:0]</td>
<td></td>
<td>BRR[19:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_RQR</td>
<td>0x10-</td>
<td>LPUART_RQR</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x14</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_ISR</td>
<td>0x18</td>
<td>LPUART_ISR</td>
</tr>
<tr>
<td></td>
<td>FIFO mode enabled</td>
<td>0x1C</td>
<td>FIFO mode enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_ICR</td>
<td>0x20</td>
<td>LPUART_ICR</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x24</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_RDR</td>
<td>0x28</td>
<td>LPUART_RDR</td>
</tr>
<tr>
<td></td>
<td>RDR[8:0]</td>
<td></td>
<td>RDR[8:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_TDR</td>
<td></td>
<td>LPUART_TDR</td>
</tr>
<tr>
<td></td>
<td>TDR[8:0]</td>
<td></td>
<td>TDR[8:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 393. LPUART register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>LPUART_CR1</td>
<td>0x00</td>
<td>LPUART_CR1</td>
</tr>
<tr>
<td></td>
<td>FIFO mode enabled</td>
<td></td>
<td>FIFO mode enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_CR2</td>
<td>0x04</td>
<td>LPUART_CR2</td>
</tr>
<tr>
<td></td>
<td>ADD[7:0]</td>
<td></td>
<td>ADD[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_CR3</td>
<td>0x08</td>
<td>LPUART_CR3</td>
</tr>
<tr>
<td></td>
<td>TXFTCFG[2:0]</td>
<td></td>
<td>TXFTCFG[2:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_BRR</td>
<td>0x0C</td>
<td>LPUART_BRR</td>
</tr>
<tr>
<td></td>
<td>BRR[19:0]</td>
<td></td>
<td>BRR[19:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_RQR</td>
<td>0x10-</td>
<td>LPUART_RQR</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x14</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_ISR</td>
<td>0x18</td>
<td>LPUART_ISR</td>
</tr>
<tr>
<td></td>
<td>FIFO mode enabled</td>
<td>0x1C</td>
<td>FIFO mode enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_ICR</td>
<td>0x20</td>
<td>LPUART_ICR</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x24</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_RDR</td>
<td>0x28</td>
<td>LPUART_RDR</td>
</tr>
<tr>
<td></td>
<td>RDR[8:0]</td>
<td></td>
<td>RDR[8:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPUART_TDR</td>
<td></td>
<td>LPUART_TDR</td>
</tr>
<tr>
<td></td>
<td>TDR[8:0]</td>
<td></td>
<td>TDR[8:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Refer to Section 2.3: Memory organization for the register boundary addresses.
55 Serial peripheral interface (SPI)

55.1 Introduction

The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The (SPI) interface supports a half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and is capable of operating in multi slave or multi master configurations. In case of master configuration it provides the communication clock (SCK) to the external slave device. The slave select signal can be provided by the master and accepted by the slave optionally, too. The Motorola data format is used by default, but some other specific modes are supported as well.
55.2 **SPI main features**

- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4-bit to 32-bit data size selection
- Multi master or multi slave mode capability
- Dual clock domain, separated clock for the peripheral kernel which can be independent of PCLK
- 8 master mode baud rate prescalers up to kernel frequency/2
- Slave mode frequency up to kernel frequency/2
- Protection of configuration and setting
- Hardware or software management of SS for both master and slave
- Adjustable minimum delays between data and between SS and data flow
- Configurable SS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transaction to control SS and CRC
- Dedicated transmission and reception flags with interrupt capability
- Slave’s transmission and/or reception capability in Stop mode (no clock provided to the peripheral) with wake up
- SPI Motorola and TI formats support
- Hardware CRC feature can secure communication at the end of transaction by:
  - Adding CRC value at Tx mode
  - Automatic CRC error checking for Rx mode
- Error detection with interrupt capability in case of data overrun, CRC error, data underrun at slave, mode fault at master
- Two 16x or 8x 8-bit embedded Rx and TxFIFOs with DMA capability
- Programmable number of data in transaction
- Configurable FIFO thresholds (data packing)
- Configurable behavior at slave underrun condition (support of cascaded circular buffers)

55.3 **SPI implementation**

<table>
<thead>
<tr>
<th>SPI modes/features</th>
<th>SPI2S1</th>
<th>SPI2S2</th>
<th>SPI2S3</th>
<th>SPI4</th>
<th>SPI5</th>
<th>SPI2S6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rx &amp; TxFIFO size (N) [x 8-bit]</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Maximum configurable data size [bits]</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>I2S feature</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
55.4 SPI functional description

55.4.1 SPI block diagram

The SPI allows a synchronous, serial communication between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main elements of SPI and their interactions are shown in the following block diagram at Figure 577.

Figure 577. SPI2S block diagram

The simplified scheme of Figure 577 shows three fully independent clock domains:

- The `spi_pclk` clock domain,
- The `spi_ker_ck` kernel clock domain,
- The serial interface clock domain,

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user has to consider a ratio compatible with the data flow speed in order to avoid any data underrun or overrun events only.
The `spi_pclk` clock signal feeds the peripheral bus interface. It has to be active when it accesses to the SPI registers are required.

The SPI working in slave mode handles data flow using the serial interface clock derived from the external SCK signal provided by external master SPI device. That is why the SPI slave is able to receive and send data even when the `spi_pclk` and `spi_ker_ck` clock signals are inactive.

This is not the case for the SPI master as it needs an active `spi_ker_ck` kernel clock coming from the RCC to feed the clock generator at least. On the other side, a specific slave logic working within the serial interface clock domain needs some additional traffic to be setup correctly (e.g. when underrun or overrun is evaluated). This cannot be done when the bus becomes into idle. At specific case the slave even requires the clock generator working (see Section 55.5.1: TI mode).

**55.4.2 SPI signals**

Four I/O pins are dedicated to SPI communication with external devices.

- **MISO**: Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI**: Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK**: Serial Clock output pin for SPI masters and input pin for SPI slaves.
- **SS**: Slave select pin. Depending on the SPI and SS settings, this pin can be used to either:
  - Select an individual slave device for communication
  - Synchronize the data frame or
  - Detect a conflict between multiple masters

See Section 55.4.6: Multi-master communication for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires: one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management. The functionality between MOSI and MISO pins can be inverted in any SPI mode (see the IOSWP bit at SPI_CFG2 register).

**55.4.3 SPI communication general aspects**

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software SS management) or 3/4 wires (with hardware SS management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slave(s) for communication by SS line when it is managed by HW. The data between the master and the slave, flow on the MOSI and/or MISO lines.

**55.4.4 Communications between one master and one slave**

The communication flow may use one of 3 possible modes: full-duplex (3 wires), half-duplex (2 wires) or simplex (2 wires). The SS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the SS
signal can be helpful at this configuration to synchronize the data flow and it is used by default at some specific SPI modes (e.g. TI mode).

**Full-duplex communication**

By default, the SPI is configured for full-duplex communication (bits COMM[1:0]=00 in the SPI_CFG2 register). In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During the SPI communication, the data are shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

**Figure 578. Full-duplex single master/ single slave application**

1. The SS pin is configured at output mode at master. The pins can be left unconnected then SS is managed by software internally on both master and slave side.

**Half-duplex communication**

The SPI can communicate in half-duplex mode by setting COMM[1:0]=11 in the SPI_CFG2 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI has to be disabled when changing direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.

**Figure 579. Half-duplex single master/ single slave application**
1. The SS pin is configured at output mode at master. The pins can be left unconnected then SS is managed by software internally on both master and slave side.

2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs.

3. A critical situation can happen when communication direction is changed not synchronously between two nodes working at bidirectional mode and new transmitter accesses the common data line while former transmitter still keeps an opposite value on the line (the value depends on SPI configuration and communicated data). Both nodes can fight with opposite outputs levels on the line temporary till next node changes its direction setting correspondingly, too. It is suggested to insert serial resistance between MISO and MOSI pins at this mode to protect the outputs and limit the current blowing between them at this situation.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pins pair is not used for communication and can be used as standard GPIOs.

• Transmit-only mode: COMM[1:0]=01
  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is on-going.
  The slave in transmit only mode sends data as long as it receives a clock on the SCK pin and the SS pin (or SW managed internal signal) is active (see 55.4.6: Multi-master communication).

• Receive-only mode: COMM[1:0]=10
  In master mode, the MOSI output is disabled and may be used as GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by SW explicitly requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the RxFIFO is full, when the MASRX bit in the SPI_CR1 is set.
  In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see 55.4.6: Multi-master communication). Received data events appear depending on the data buffer configuration.

Note: At whatever master and slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register. (This bit may only be modified when the SPI is disabled). Any simplex communication can be replaced by a variant of the half duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled, while the HDDIR bit is never changed).
55.4.5 **Standard multi-slave communication**

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see Figure 581). The master must select one of the slaves individually by pulling low the GPIO connected to the slave SS input (only one slave can control data on common MISO line at time). When this is done, a communication between the master and the selected slave is established. Except the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.
Figure 581. Master and three independent slaves at star topology

1. SS pin is not used on master side at this configuration. It has to be managed internally (SSM=1, SSI=1) to prevent any MODF error.

2. If the application cannot ensure that no more than a single SS active signal is provided by the master at time, it is better to configure the MISO pins into open drain configuration with an external pull-up at MISO line to prevent any conflict between interconnected outputs of the slaves on the line. Else the push-pull configuration can be applied without an extra resistor for the slaves. (see Section 11.3.7: I/O alternate function input/output on page 507.)

The master can handle the SPI communication with all the slaves in time when a circular topology is applied (see Figure 582). All the slaves behave like simple shift registers applied at serial chain under common slave select and clock control. All the information is shifted simultaneously around the circle while returning back to the master. Sessions have fixed the length where the number of data frames transacted by the master is equal to the number of slaves. Then when a first data frame is transacted in the chain, the master just sends information dedicated for the last slave node in the chain via the first slave node input while the first information received by the master comes from the last node output at this time. Correspondingly, the lastly transacted data finishing the session is dedicated for the first slave node while its firstly outgoing data just reaches the master input after its circling around the chain passing through all the other slaves during the session. The data format
configuration and clock setting has to be the same for all the nodes in the chain at this topology. As the receive and transmit shift registers are separated internally, a trick with intentional underrun has to be applied at the TxFIFO slaves when information is transacted between the receiver and the transmitter by hardware. In this case, the transmission underrun feature is configured at a mode repeating lastly received data frame (UDRCFG[1:0]=01). A session can start optionally with a single data pattern written into the TxFIFO by each slave (usually slave status information is applied) before the session starts. In this case the underrun happens in fact after this first data frame is transacted (underrun detection has to be set at end of data transaction at slaves UDRDET[1:0]=01). To be able to clear the internal underrun condition immediately and restart the session by the TxFIFO content again, the user has to disable and enable the SPI between sessions and fill the TxFIFO by a new single data pattern (to overcome the propagating delay of the clearing raised in case the underrun is cleared in a standard way by the UDRC bit).

Figure 582. Master and three slaves at circular (daisy chain) topology

1. Underrun feature is used at slaves at this configuration when slaves are able to transmit data received previously into the Rx shift register once their TxFIFOs become empty.
55.4.6 **Multi-master communication**

Unless the SPI bus is not designed for a multi-master capability primarily, the user can use build in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the SS pin is used configured at hardware input mode. The connection of more than two SPI nodes working at this mode is impossible as only one node can apply its output on a common data line at time.

When nodes are non active, both stay at slave mode by default. Once one node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is completed, the active slave select signal is released and the node mastering the bus temporary returns back to passive slave mode waiting for next session start.

If potentially both nodes raised their mastering request at the same time a bus conflict event appears (see mode fault MODF event). Then the user can apply some simple arbitration process (e.g. to postpone next attempt by predefined different time-outs applied at both nodes).

**Figure 583. Multi-master application**

1. The SS pin is configured at hardware input mode at both nodes. Its active level enable the MISO line output control as passive node is configured as a slave.

55.4.7 **Slave select (SS) pin management**

In slave mode, the SS works as a standard ‘chip select’ input and lets the slave communicate with the master. In master mode, the SS can be used either as an output or an input. As an input it can prevent a multi master bus collision, and as an output it can drive a slave select signal of a single slave. The SS signal can be managed internally (software management of the SS input) or externally when both the SS input and output are associated with the SS pin (hardware SS management). The user can configure which level of this input/output external signal (present on the SS pin) is considered as active one by the SSIOP bit setting. While low level is considered as active internally SSIOP=1 setting can invert this logic for external world.
The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software SS management (SSM = 1):** in this configuration, slave select information is driven internally by the SSI bit value in the register SPI_CR1. The external SS pin is free for other application uses (as GPIO or other alternate function).

- **Hardware SS management (SSM = 0):** in this case, there are two possible configurations. The configuration used depends on the SS output configuration (SSOE bit in register SPI_CFG2).
  - **SS output enable (SSOE = 1):** this configuration is only used when the MCU is set as master. The SS pin is managed by the hardware.
    a) When SSOM = 0 and SP = 000, the SS signal is driven to the active level as soon as the master transfer starts (CSTART=1) and it is kept active until its EOT flag is set or the transmission is suspended.
    b) When SP = 001, a pulse is generated as defined by the TI mode.
    c) When SSOM=1, SP=000 and MIDI>1 the SS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (1 to 14).
  - **SS output disable (SSM=0, SSOE = 0):**
    a) If the microcontroller is acting as the master on the bus, this configuration allows multi master capability. If the SS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER=0).
    b) In slave mode, the SS pin works as a standard ‘chip select’ input and the slave is selected while the SS line is at its active level.

**Note:**

The purpose of automatic switching into slave mode at mode fault condition is to avoid the possible conflicts on data and clock line. The SPE is not automatically reset, as this would automatically flush both RX and TxFIFOs and current data may be lost. Following the MODF event, the SW must correctly manage the FIFO read/flush and correctly re-program the SPI configuration for taking over the slave role in the system.

**Note:**

When the SPI slave is enabled at the hardware SS management mode, all the traffics are ignored even in case of the SS is found at active level till the slave detects a start of the SS signal (its transaction from non-active to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when the external SS pin is fixed. There is no such protection at the SS software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.
When a hardware output SS control is applied (SSM=0, SSOE=1), by configuration of MIDI[3:0] and MSSI[3:0] bit fields the user can control timing of the SS signal between data frames and insert an extra delay at begin of every transaction (to separate the SS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see Figure 585: Data flow timing control (SSOE=1, SSOM=0, SSM=0)).

2. CPHA=0, CPOL=0, SSOP=0, LSBFRST=0.

Additionally, bit SSOM=1 setting invokes specific mode which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] has to be set greater than one SPI period). Some configuration examples are shown at Figure 586: SS interleaving pulses between data (SSOE=1, SSOM=1, SSM=0).
### Serial peripheral interface (SPI)

#### Figure 586. SS interleaving pulses between data (SSOE=1, SSOM=1, SSM=0)

<table>
<thead>
<tr>
<th>Case</th>
<th>CPHA</th>
<th>CPOL</th>
<th>SSOP</th>
<th>LSBFRST</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>II.</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>III.</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>IV.</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **CPHA**: CPHA is used to select between CPHA=0 and CPHA=1 mode.
- **CPOL**: CPOL is used to select between CPOL=0 and CPOL=1 mode.
- **SSOP**: SSOP is used to select between SSOP=0 and SSOP=1 mode.
- **LSBFRST**: LSBFRST is used to select between LSBFRST=0 and LSBFRST=1 mode.

- **MIDI[3:0]**: MIDI[3:0] is used to select between different data modes.
- **SCK**: SCK is the clock input.
- **MOSI**: MOSI is the master output slave input.
- **MISO**: MISO is the master input slave output.
- **DSIZE[4:0]**: DSIZE[4:0] is the data size.
- **MSB**: MSB is the most significant bit.
- **LSB**: LSB is the least significant bit.

---

**Diagram Details:**
- **I. CPHA=0, CPOL=0, SSOP=0, LSBFRST=0**: SSOE=1, SSOM=1, SSM=0.
- **II. CPHA=1, CPOL=0, SSOP=0, LSBFRST=0**: SSOE=1, SSOM=1, SSM=0.
- **III. CPHA=0, CPOL=1, SSOP=1, LSBFRST=1**: SSOE=1, SSOM=1, SSM=0.
- **IV. CPHA=1, CPOL=1, SSOP=1, LSBFRST=1**: SSOE=1, SSOM=1, SSM=0.
2. \text{SS} interleaves between data when \text{MIDI}[3:0]>1.

### 55.4.8 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slave devices must follow the same communication format and be synchronized correctly.

#### Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPI\_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data are being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

\textit{Figure 587}, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

\textit{Note:} Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit. The idle state of SCK must correspond to the polarity selected in the SPI\_CFG2 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).
Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFRST bit in SPI_CFG2 register. The data frame size is chosen by using the DSIZE[4:0] bits. It can be set from 4-bit up to 32-bit length and the setting applies for both transmission and reception. When the SPI_TXDR/SPI_RXDR registers are accessed, data frames are always right-aligned into either a byte (if the data fit into a byte), a half-word or a word (see Figure 588).

If the access is a multiple of the minimum data size needed for a single data frame, 2 or 4 data frames are packed into the register. During communication, only bits within the data frame are clocked and transferred.
Figure 588. Data alignment when data size is not equal to 8-bit, 16-bit or 32-bit

<table>
<thead>
<tr>
<th>DSIZE &lt;= 8-bits</th>
<th>DSIZE &lt;= 9-bits</th>
<th>DSIZE &lt;= 17-bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>data is right aligned on byte</td>
<td>data is right aligned on half-word</td>
<td>data is right aligned on word</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7</th>
<th>4</th>
<th>3</th>
<th>0</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>0</th>
<th>31</th>
<th>27</th>
<th>26</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7</th>
<th>4</th>
<th>3</th>
<th>0</th>
<th>15</th>
<th>13</th>
<th>12</th>
<th>0</th>
<th>31</th>
<th>27</th>
<th>26</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Note: The minimum data length is 4 bits. If a data length of less than 4 bits is selected, it is forced to an 4-bit data frame size.

55.4.9 Configuration of SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication has to be initialized, perform these steps:

1. Write the proper GPIO registers: Configure GPIO for MOSI, MISO and SCK pins.
2. Write to the SPI_CFG1 and SPI_CFG2 registers to set up proper values of all not reserved bits and bit fields included there with next exceptions:
   a) SSOM, SSOE, MBR[2:0], MIDI[3:0] and MSSI[3:0] are required at master mode only, the MSSI bits take effect when SSOE is set, MBR setting is required for slave at TI mode, too
   b) UDRDET[1:0] and UDRCFG[1:0] are required at slave mode only,
   c) CRCSIZE[4:0] is required if CRCEN is set,
   d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP and SSM are not required at TI mode.
   e) Once the AFCNTR bit is set at SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless the peripheral enable so any later configurations changes of the SPI_CFG1 and SPI_CFG2 registers can affect level of signals at these pins.
   f) The I2SMOD bit at SPI_I2SCFGR register has to be kept cleared to prevent any unexpected influence of occasional I2S configuration.
3. Write to the SPI_CR2 register to select length of the transfer, if it is not known TSIZE has to be programmed to zero.
4. Write to SPI_CRCPOLY and into TCRCINI, RCRCINI and CRC33_17 bits at SPI2S_CR1 register to configure the CRC polynomial and CRC calculation if needed.
5. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see chapter Communication using DMA).
6. Program the IOLOCK bit in the SPI_CFG1 register if the configuration protection is required (for safety).
55.4.10 Procedure for enabling SPI

It is recommended to configure and enable the SPI slave before the master sends the clock but there is no impact if the configuration and enabling procedure is done while a traffic is ongoing on the bus. The data register of the slave transmitter must contain data to be sent before the master starts its clocking. The SCK signal must be settled to idle state level corresponding to the selected polarity before the SPI slave is selected by SS else following transaction may be desynchronized.

When the SPI slave is enabled at the hardware SS management mode all the traffics are ignored even in case of the SS is found at active level till the slave detects a start of the SS signal (its transaction from non-active to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when external SS pin is fixed. There is no such protection at the SS software management. In this case the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master at full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see Section 55.4.14: Communication using DMA (direct memory addressing).

55.4.11 SPI data transmission and reception procedures

RxFIFO and TxFIFO

All SPI data transactions pass through the embedded FIFOs organized by bytes (N x 8-bit). The size of the FIFOs (N) is product and the peripheral instance dependent. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short or the interrupt/DMA latency is too long. Each direction has its own FIFO called TxFIFO and RxFIFO, respectively.

The handling of FIFOs depends on the data exchange mode (duplex, simplex), the data frame format (number of bits in the frame), the access size performed on the FIFO data registers (8-bit, 16-bit or 32-bit), and how data are organized at packets.

A read access to the SPI2S_RXDR register returns the oldest value stored in the RxFIFO that has not been read yet. A write access to the SPI2S_TXDR stores the written data in the TxFIFO at the end of a send queue.

A read access to the SPI2S_RXDR register must be managed by the RXP event. This flag is set by hardware when at least one complete data packet (defined as receiver threshold by FTHLV[3:0] bits at the SPI_CFG1 register) is available at the reception FIFO while reception is active. The RXP is cleared as soon as less data are available in the RxFIFO, when reading SPI2S_RXDR by software or by DMA.

The RXP triggers an interrupt if the RXPIE bit is set or a/o a DMA request if RXDMAEN is set.

Upon setting of the RXP flag, the application software performs the due number of SPI data register reads to download the content of one data packet. Once a complete data packet is downloaded, the application software checks the RXP value to see if other packets are
pending into the receive FIFO and, if so, downloads them packet by packet until the RXP reads 0. RxFIFO can store up to N data frames (for frame size =< 8-bit), N/2 data frames (for 8-bit < frame =< 16-bit), N/3 data frames (for 16-bit < frame =< 24-bit) or N/4 data frames (if data frame > 24-bit) where N is the size of the FIFO in bytes.

At the end of a reception, it may happen that some data may still be available in the RxFIFO, without reaching the FTHLV level, thus the RXP is not set. In this case, the number of remaining RX data frames in the FIFO is indicated by RXWNE and RXPLVL fields in the SPI_SR register. It happens when number of the last data received in a transfer cannot fully accomplish the configured packet size in the case transfer size and packet size are not aligned. Nevertheless the application software can still perform the standard number of reads from the RxFIFO used for the previous complete data packets without drawbacks: only the consistent data (completed data frames) are popped from the RxFIFO while redundant reads (or any uncompleted data) are reading 0. Thanks to that, the application software can treat all the data in a transfer in the same way and is off-loaded to foresee the reception of the last data in a transfer and from calculating the due number of reads to be popped from RxFIFO.

In a similar way, write access of a data frame to be transmitted is managed by the TXP event. This flag is set by hardware when there is enough space for the application software to push at least one complete data packet (defined at FTHLV[3:0] bits at SPI_CFG1 register) into the transmission FIFO while transmission is active. The TXP is cleared as soon as the TxFIFO is filled by software a/o by DMA and space currently available for any next complete data packet is lost. This can lead to oscillations of the TXP signal when data are released out from the TxFIFO while a new packet is stored frame by frame. Any write to the TxFIFO is ignored when there is no sufficient room to store at least a single data frame (TXP event is not respected), when TXTF is set or when the SPI is disabled.

The TXP triggers an interrupt if the TXPIE bit is set or a/o a DMA request if TXDMAEN is set. The TXPIE mask is cleared by hardware when the TXTF flag is set.

Upon setting of the TXP flag application software performs the due number of SPI data register writes to upload the content of one entire data packet. Once new complete data packet is uploaded, the application software checks the TXP value to see if other packets can be pushed into the TxFIFO and, if so, uploads them packet by packet until TXP reads 0 at the end of any packet load.

The number of last data in a transfer can be shorter than the configured packet size in the case when the transfer size and the packet size are not aligned. Nevertheless the application software can still perform the standard number of data register writes used for the previous packets without drawbacks: only the consistent data are pushed into the TxFIFO while redundant writes are discarded. Thanks to that, the application software can treat all the data in a transfer in the same way and is off-loaded to foresee the transmission of the last data in a transfer and from calculating the due number of writes to push the last data into TxFIFO. Just for the last data case, the TXP event is asserted by SPI once there is enough space into TxFIFO to store remaining data to complete current transfer.

Both TXP and RXP events can be polled or handled by interrupts. The DXP bit can be monitored as a common TXP and RXP event at full duplex mode.

Upon setting of the DXP flag the application software performs the due number of writes to the SPI data register to upload the content of one entire data packet for transmission, followed by the same number of reads from the SPI data register to download the content of one data packet. Once one data packet is uploaded and one is downloaded, the application software checks the DXP value to see if other packets can be pushed and popped in sequence and, if so, uploads/downloads them packet by packet until DXP reads 0.
The DXP triggers an interrupt if the DXPIE bit is set or a/o a DMA requests if TXDMAEN and RXDMAEN are set. The DXPIE mask is cleared by hardware when the TXTF flag is set.

The DXP is useful in Full-Duplex communication in order to optimize performance in data uploading/downloading, and reducing the number of interrupts required to support an SPI transfer thus minimizing the request for CPU bandwidth and system power especially when SPI is operated in Stop mode.

Another way to manage the data exchange is to use DMA (see Communication using DMA (direct memory addressing)).

If the next data is received when the RxFIFO is full, an overrun event occurs (see description of OVR flag at Section 55.5.2: SPI error flags). An overrun event can be polled or handled by an interrupt.

This may happen in slave mode or master mode (full duplex or receive only with MASRX = 0). In master receive only mode, with MASRX = 1, the generated clock stops automatically when the RxFIFO is full, therefore overrun is prevented.

Both RxFIFO and TxFIFO content is kept flushed when SPI is disabled (SPE=0).

### Sequence handling

A few data frames can be passed at single sequence to complete a message. The user can handle number of data within a message thanks to values stored into TSIZE and TSER fields. In principle, the transaction of a message starts when the SPI is enabled by setting CSTART bit and finishes when number of required data is transacted. The end of transaction controls the CRC and the hardware SS management when applied. If TSIZE is kept at zero while CSTART is set, an endless transaction is initialized (no data size control is applied). The transaction can be suspended at any time thanks to CSUSP which clears the CSTART bit.

In master mode, the user can extend the number of data within the current session. When the number of data programmed into TSIZE is transacted and if TSER contains a non-zero value, the content of TSER is copied into TSIZE, and TSER value is cleared automatically. The transaction is then extended by a number of data corresponding to the value reloaded into TSIZE. The EOT event is not raised in this case as the transaction continues. After the reload operation, the TSERF flag is set and an interrupt is raised if TSERFIE is set. The user can write the next non-zero value into TSER before the next reload occurs, so an unlimited number of data can be transacted while repeating this process.

When any data extension is applied, it always starts by aligned data packet. That is why it is suggested to keep number of data to be extended always aligned with packet size else the last data packet just before the extension is applied has to be handled as an incomplete one (see data packing chapter). If overall number of data is not aligned, the user must implement the rest not aligned number of data into TSER just at the last extension cycle and then handle the last incomplete packet of data standardly within EOT event handler.

For example, if the user wants to transfer 23 bytes while applies data number extension at configuration of 8-bit data size, data packet set to 4 data and 32-bit access to FIFO is used then whatever next sequence is correct

- TSIZE=16 TSER=7;
- TSIZE=12 TSER=8; last extension TSER=3;

As the last not aligned MSB byte is ignored just within the last (6th) access of the FIFO.
When a not aligned sequence is applied for data to be extended like at the following cases

- TSIZE=15 TSER=8 or
- TSIZE=8 TSER=7; last extension TSER=8;

The MSB byte is ignored within the 4th access of the FIFO while the other accesses handle always 4 data at the FIFO.

When the transmission is enabled, a sequence begins and continues while any data is present in the TxFIFO of the master. The clock signal is provided permanently by the master until TxFIFO becomes empty, then it stops, waiting for additional data.

In receive-only modes, half duplex (COMM[1:0]=11, HDDIR=0) or simplex (COMM[1:0]=10) the master starts the sequence when SPI is enabled and transaction is released by setting the CSTART bit. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled/suspended by the master. The master receives data frames permanently up to this moment. The reception can be suspended either by SW control, writing 1 to the CSUSP bit in the SPI_CR1 register, or automatically when MASRX=1 and RxFIFO becomes full. The reception is automatically stopped also when the number of frames programmed in TSIZE and TSER fields of the SPI_CR2 register has been completed.

In order to disable the master receive only mode, the SPI must be suspended at first. When the SPI is suspended, the current frame is completed, before changing the configuration.

**Caution:** If SPE is written to 0 at master, while reception is ongoing without any suspending, the clock is stopped without completing the current frame, and the RxFIFO is flushed.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays by MIDI[3:0] bits setting or provide an initial delay by setting MSS[1:0] which postpones any transaction start to give slave sufficient room for preparing data. Be aware data from the slave are always transacted and processed by the master even if the slave could not prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are short, FIFO is accessed by bytes and the SPI bus rate is high.

In order to add some SW control on the SPI communication flow from a slave transmitter node, a specific value written in the SPI_UDRDR (SPI Underrun Data Register) may be used. On slave side, when TxFIFO becomes empty, this value is sent out automatically as next data and may be interpreted by SW on the master receiver side (either simply dropped or interpreted as a XOFF like command, in order to suspend the master receiver by SW).

At multi-slave star topology, a single slave can be only enabled for the output data at a time. The slave just selected for the communication with the master needs to detect a change of its SS input into active level before the communication with the master starts. In a single slave system it is not necessary to control the slave with SS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. The SS can be managed by both software and hardware (Section 55.4.7: Slave select (SS) pin management).
55.4.12 Procedure for disabling the SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph.

At the master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Otherwise, ongoing transactions may be corrupted in this case.

In slave mode, the SPI communication can continue when the spi_pclk and spi_ker_ck clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The spi_pclk can generally be stopped by setting the system into STOP mode. Refer to the RCC section for further information.

The master in full duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. TXC flag can be polled (or interrupt enabled with EOTIE=1) in order to wait for the last data frame to be sent.

When the master is in any receive only mode, in order to stop the peripheral, the SPI communication must be first suspended, by setting CSUSP to 1.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

When SPI is disabled, Rx FIFO is flushed. To prevent losing unread data, the user has to ensure that Rx FIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the RXP, RXWNE and RXPLVL fields in the SPI_SR register).

The standard disable procedure is based on polling EOT and/or TXC status to check if a transmission session is (fully) completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When the SS signal is managed by software and the master has to provide proper end of SS pulse for slave, or
- When transaction streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

The correct disable procedure in master mode, except when receive only mode is used, is:

1. Wait until TXC=1 and/or EOT=1 (no more data to transmit and last data frame sent).
   When CRC is used, it is sent automatically after the last data in the block is processed. TXC/EOT is set when CRC frame is completed in this case. When a transmission is suspended the software has to wait till CSTART bit is cleared.
2. Read all Rx FIFO data (until RXWNE=0 and RXPLVL=00)
3. Disable the SPI (SPE=0).

The correct disable procedure for master receive only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP=1)
2. Wait until SUSP=1 (the last data frame is processed) if receive flow is suspended.
3. Read all Rx FIFO data (until RXWNE=0 and RXPLVL=00)
4. Disable the SPI (SPE=0).

In slave mode, any ongoing data are lost when disabling the SPI.
55.4.13 Data packing

From user point of view there are two ways of data packing which can overlay each other:

- Type of access when data are written to TxFIFO or read from RxFIFO
  *Multiple data can be pushed or fetched effectively by single access if data size is considerably less than access performed upon SPI2S_TXDR or SPI2S_RXDR registers.*

- Number of data to be handled during the single software service
  *It is convenient to group data into packets and cumulate the FIFO services overall the data packet content exclusively instead of handling data frame by frame separately.*
  *The user can define packets by FIFO threshold settings. Then all the FIFO occupancy events are related to that threshold level while required services are signaled by proper flags with interrupt and/or wake up capabilities.*

When the data frame size fits into one byte (less than or equal to 8 bits), the data packing is used automatically when any read or write 16-bit or 32-bit access is performed on the SPI2S_RXDR/SPI2S_TXDR register. The multiple data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other data stored in the MSB. *Figure 589* provides an example of data packing mode sequence handling. While DSIZE[3:0] is configured to 4-bit there, two or four data frames are written in the TxFIFO after the single 16-bit or 32-bit access the SPI2S_TXDR register of the transmitter.

When the data frame size is between 9-bit and 16-bit, data packing is used automatically when a 32-bit access is done. the least significant half-word is used first. (regardless of the LSBFRST value)

This sequence can generate two or four RXP events in the receiver if the RxFIFO threshold is set to 1 frame (and data is read on a frame basis, unpacked), or it can generate a single RXP event if the FTHLV[3:0] field in the SPI_CFG1 register is programmed to a multiple of the frames to be read in a packed mode (16-bit or 32-bit read access).

The data are aligned in accordance with *Figure 588: Data alignment when data size is not equal to 8-bit, 16-bit or 32-bit.* The valid bits are performed on the bus exclusively. Unused bits are not cared at transmitter while padded by zeros at receiver.

When short data frames (<8-bit or < 16-bit) are used together with a larger data access mode (16-bit or 32-bit), the FTHLV value must be programmed as a multiple of the number of frames/data access (i.e. multiple of 4 if 32-bit access is used to up to 8-bit frames or multiple of 2 if 16-bit access is used to up to 8-bit frames or 32-bit access to up to 16-bit frames.).

The RxFIFO threshold setting must always be higher than the following read access size, as spurious extra data would be read otherwise.

The FIFO data access less than the configured data size is forbidden. One complete data frame has to be always accessed at minimum.

A specific problem appears if an incomplete data packet is available at FIFO: less than 4x8-bit frames or one single 16-bit frame is available.

There are two ways of dealing with this problem:
A. without using TSIZE field

On transmitter side, writing the last data frame of any odd sequence with an 8-bit/16-bit access to SPI2S_TXDR is enough.

On receiver side, the remaining data may be read by any access. Any extra data read are padded with zeros. Polling the RXWNE and RXPLVL may be used to detect when the RX data are available in the RxFIFO. (a time out may be used at system level in order to detect the polling)

B. using the TSIZE field

On transmitter side, the transaction is stopped by the master when it faces EOT event. In reception, the RXP flag is not set when EOT is set. In the case when the number of data to be received (TSIZE) is not a multiple of packet size, the number of remaining data is indicated by the RXWNE and RXPLVL fields in the SPI_SR register. The remaining data can be read by any access. Any extra read is padded by zeros.

Figure 5.89. Packing data in FIFO for transmission and reception

1. DSIZE[3:0] is configured to 4-bit, data is right aligned, valid bits are performed only on the bus, their order depends on LSBFRST, content of LSB byte goes first on the bus.

55.4.14 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bits in the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a series of DMA requests is triggered each time TXP is set to 1. The DMA then performs series of writes to the SPI2S_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set to 1. The DMA then performs series of reads from the SPI2S_RXDR register. When EOT is set at the end of transaction and last data packet is incomplete then DMA request is activated automatically in according with RXWNE and RXPLVL[1:0] setting to read rest of data.

When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.
If the SPI is programmed in receive only mode, UDR is never set.

If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel.

If the SPI is programmed in transmit only mode, RXP and OVR are never set.

If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set at SPI2C_SR register), the TXC flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or before disabling the spi_pclk in master mode. The software must first wait until EOT=1 and/or TXC=1.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI_CFG1 register, if DMA Rx is used.
2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CFG1 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order:

1. Disable DMA request for Tx and Rx in the DMA registers, if the DMA issued.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI_CFG1 register, if DMA Tx and/or DMA Rx are used.

Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel.

If the DMA channel PSIZE value is equal to 16-bit or 32-bit and SPI data size is less than or equal to 8-bit, then packing mode is enabled. Similarly, If the DMA channel PSIZE value is equal to 32-bit and SPI data size is less than or equal to 16-bit, then packing mode is enabled. The DMA then automatically manages the write operations to the SPI2S_TXDR register.

Regardless data packing mode is used and the number of data to transfer is not a multiple of the DMA data size (16-bit or 32-bit) while the frame size is smaller, DMA completes the transfer automatically in according with the TSIZE field setting.

Alternatively, last data frames may be written by software, in the single/unpacked mode.

To configure any DMA data access less than the configured data size is forbidden. One complete data frame has to be always accessed at minimum.
55.5 **SPI specific modes and control**

55.5.1 **TI mode**

By specific setting of the SP[2:0] bit field at the SPI_CFG2 register the SPI can be configured to be compliant with TI protocol. The SCK and SS signals polarity, phase and flow as well as the bits order are fixed so the setting of CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP and SSM is not required when the SPI is at TI mode configuration. The SS signal synchronizes the protocol by pulses over the LSB data bit as it is shown at the Figure 590: TI mode transfer.

![Figure 590. TI mode transfer](image)

In slave mode, the clock generator is used to define time when the slave output at MISO pin becomes to HiZ when the current transaction finishes. The master baud setting is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become HiZ (TRELEASE) depends on internal re-synchronization, too. It is given by formula:

\[
\frac{T_{baud}}{2} + 2 \times T_{com} < T_{release} < \frac{T_{baud}}{4} + 4 \times T_{com}
\]

If the slave detects misplaced SS pulse during data transaction the TIFRE flag is set.

55.5.2 **SPI error flags**

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the corresponding Interrupt Enable bit.

**Overrun flag (OVR)**

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RxFIFO).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all data transmitted subsequently are lost. OVR flag triggers an interrupt if OVRIE bit is set. Clearing the OVR bit is done by a writing 1 to the OVRC bit in the SPI_IFCR. To prevent any
next overrun event the clearing must be done after RxFIFO is emptied by software reads. At master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

**Underrun flag (UDR)**

At a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO at the moment specified by UDRDET bits. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Once the underrun is captured next provided data for transmission depends on the UDRCFG bits. The slave can provide out either data stored lastly to its TxFIFO or the data received previously from the master or a constant pattern stored by the user at the UDRDR register. The second configuration can be used at circular topography structure (see Figure 582). Standard transmission is re-enabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. The user must write some data into TxFIFO prior clearing UDR flag to prevent any next underrun condition occurrence capture.

The data transacted by slave is unpredictable especially when the transaction starts or continues while TxFIFO is empty and underrun condition is either not yet captured or just cleared. Typically, this is the case when UDRDET[1:0]=00 or SPI is just enabled or when a transaction with a defined size just starts. First bits can be corrupted in this case, as well, when slave software writes first data into the empty TxFIFO too close prior the data transaction starts (propagation of the data into TxFIFO takes few APB clock cycles). If the user cannot ensure to write data into the empty TxFIFO in time the UDRDET[1:0]=00 setting must be avoided.

To handle the underrun control feature correctly the user must avoid next critical encroachments especially
- Any fill of empty TxFIFO when master starts clocking (at UDRDET[1:0]=00 especially)
- Any clear of UDR flag while TxFIFO is empty
- Any setting of UDRDET[1:0]=00 together with UDRCFG[1:0]=10
- Any setting of UDRDET[1:0]=10 when underrun must be detected after each data frame while SS signal does not toggle between the frames
- Any setting of UDRDET[1:0]=10 while SS is managed by software

**Note:** The hardware propagation of an UDR flag change needs additional traffic on the bus. It always takes 3 SPI clock cycles after the event happen (underrun captured by hardware or the UDR flag cleared by software).

**Mode fault (MODF)**

Mode fault occurs when the master device has its internal SS signal (SS pin in SS hardware mode, or SSI bit in SS software mode) pulled low. This automatically affects the SPI interface in the following ways:
- The MODF bit is set and an SPI interrupt is generated if the MODFIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MASTER bit is cleared, thus forcing the device into slave mode.

MODF is cleared by writing 1 to the MODFC bit in the SPI_IFCR.

To avoid any multiple slave conflicts in a system comprising several MCUs, the SS pin must be pulled to its non-active level before re-enabling the SPI, by setting the SPE bit.
As a security, hardware does not allow the SPE bit to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multi master conflict.

A correct SW procedure when master overtakes the bus at multi master system must be the following one:
- Switch into master mode while SSOE=0
  (potential conflict can appear when another master occupies the bus. MODF is raised in this case which prevents any next node switching into master mode)
- Put GPIO pin dedicated for another master SS control into active level
- Perform data transaction
- Put GPIO pin dedicated for another master SS control into non active level
- Switch back to slave mode

**CRC error (CRCE)**

This flag is used to verify the validity of the value received when the CRCEN bit in the SPI_CFG1 register is set. The CRCE flag in the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if RCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit in the SPI_IFCR.

**TI mode frame format error (TIFRE)**

A TI mode frame format error is detected when an SS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the SS pulse is ignored, and the SPI waits for the next SS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of few data bytes.

The TIFRE flag is cleared by writing 1 to the TIFREC bit in the SPI_IFCR. If the TIFREIE bit is set, an interrupt is generated on the SS error detection. As data consistency is no longer guaranteed, communication must be re-initiated by software between master and slave.

### 55.5.3 CRC computation

Two separate 33-bit or two separate 17-bit CRC calculators are implemented in order to check the reliability of transmitted and received data. The SPI offers any CRC polynomial length from 5 to 33 bits when maximum data size is 32-bit and from 5 to 17 bits for the peripheral instances where maximum data size is limited to 16-bit. The length of the polynomial is defined by the most significant bit of the value stored at the CRCPOLY register. It has to be set greater than data frame length defined at DSIZE field. When maximum data size is applied, the CRC33_17 bit has to be set additionally to define the most significant bit of the polynomial string while keep its size always greater than data. The CRCSIZE field in the SPI_CFG1 then defines how many the most significant bits from CRC calculation registers are transacted and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size.

**CRC principle**

The CRC calculation is enabled by setting the CRCEN bit in the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial
defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

**CRC transfer management**

Communication starts and continues normally until the last data frame has to be sent or received in the SPI_DR register.

The length of the transfer has to be defined by TSIZE and TSER. When the desired number of data is transacted, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

TSIZE cannot be set to 0xFFFF value if CRC is enabled. A correct way of sending e.g. 65535 data with CRC is to set:

- TSIZE= 0xFFFE and TSER=1 when data packet is configured to keep one data respective
- TSIZE= 0xFFFFC and TSER=3 when data packet keeps 4 data (to ensure the TSIZE value aligned with packet size when its extension is applied).

In transmission, the CRC computation is frozen during CRC transaction and the TXCRC is transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when desired number of data is transacted. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is completed, an automatic check is performed comparing the received CRC value and the value calculated in the SPIx_RXCRC register. Software has to check the CRCERR flag in the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing 1 to the CRCERRC.

The user takes no care about any flushing redundant CRC information, it is done automatically.

**Resetting the SPIx_TXCRC and SPIx_RXCRC values**

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode in order to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI at SPI2S_CR1 register.

The CRC values are reset when the SPI is disabled.
55.6 Low-power mode management

The SPI has advanced low-power mode functions allowing it to transfer properly data between the FIFOs and the serial interface even when the spi_pclk clock is disabled.

In master mode the spi_ker_ck kernel clock is needed in order to provide the timings of the serial interface.

In slave mode, the spi_ker_ck clock can be removed as well during the transfer of data between the FIFOs and the serial interface. In this mode the clock is provided by the external SPI device.

When the spi_pclk clock is gated, (and the spi_ker_ck clock as well if the SPI is in slave), the SPI provides a wakeup event signal (spi_wkup) if a specific action requiring the activation of the spi_pclk clock is needed, such as:

- To fill-up the TxFIFO,
- To empty the RxFIFO,
- Other signaling: end of transfer, errors...

The generation of spi_ker_ck and spi_pclk clock are controlled by the RCC block according to register settings and the processors modes. Refer to the RCC section for details.

The application shall acknowledge all pending interrupts events before switching the SPI to low-power mode (i.e. removing spi_pclk).

The Figure 591 shows an example of the clock handling when the SPI2S is working in low-power mode. The example is given for a transmit mode.

In master mode the spi_ker_ck clock is required for the timing generation.

The Figure 591 shows two kinds of supported scenarios for the handling of the spi_ker_ck kernel clock in slave mode:

- In most of the slave modes, the spi_ker_ck kernel clock can be disabled,
- In some products, the spi_ker_ck kernel clock activation may follow the system state.
The figure clearly shows that the **spi_pclk** must be provided to the SPI2S, when data need to be transferred from the memory to the SPI2S TxFIFO. Here is the description of the most important steps:

- **Step 1**
  The TxFIFO level goes below the programmed threshold, this event (TXP) activates the **spi_wkup** signal. This signal is generally used to wake-up the system from low-power mode, and thus to activate the bus clock (**spi_pclk**).

- **Step 2**
  When **spi_pclk** is activated, the **spi_it** is also activated, and the product is ready to fill-up the TxFIFO either by DMA or by software. Note as well that for some product the system wake-up automatically enables the **spi_ker_ck** kernel clock as well.

- **Step 3**
  When the amount of empty locations in the TxFIFO is less than FTHLV, then the **spi_wkup** and **spi_it** signals are deactivated, but the fill-up of the TxFIFO may continue. Note that **spi_wkup** falling edge is aligned with the serial interface clock domain, and the falling edge of the **spi_it** is aligned with the **spi_pclk** clock domain.

- **Step 4**
  The fill-up of the TxFIFO is completed; the software can switch the system back to low-power mode until the next **spi_wkup** occurs.
55.7 SPI wakeup and interrupts

Table 395 gives an overview of the SPI events capable to generate interrupt events (spi_it). Some of them feature wake-up from low-power mode capability additionally (spi_wkup).

Most of them can be enabled and disabled independently while using specific interrupt enable control bits.

Table 395. SPI wakeup and interrupt requests

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag (1)</th>
<th>Enable Control bit</th>
<th>Event clear method</th>
<th>Interrupt/Wakeup activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)</td>
<td>TXF</td>
<td>TXPIE</td>
<td>TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations</td>
<td>spi_it</td>
</tr>
<tr>
<td>Data received in Rx FIFO (one data packet available - FIFO threshold)</td>
<td>RXF</td>
<td>RXPIE</td>
<td>RXF cleared by hardware when RxFIFO contains less than FTHLV samples</td>
<td>spi_it</td>
</tr>
<tr>
<td>Both TXP and RXP active</td>
<td>DXF</td>
<td>DXPIE</td>
<td>When TXP or RXP are cleared</td>
<td>spi_it</td>
</tr>
<tr>
<td>Transmission Transfer Filled</td>
<td>TTXF</td>
<td>TTXFIE</td>
<td>Writing TTXFC to 1</td>
<td>spi_it</td>
</tr>
<tr>
<td>Underrun</td>
<td>UDR</td>
<td>UDRIE</td>
<td>Writing UDRC to 1</td>
<td>spi_it</td>
</tr>
<tr>
<td>Overrun</td>
<td>OVR</td>
<td>OVRIE</td>
<td>Writing OVRC to 1</td>
<td>spi_it</td>
</tr>
<tr>
<td>CRC Error</td>
<td>CRCE</td>
<td>CRCEIE</td>
<td>Writing CRCEC to 1</td>
<td>spi_it</td>
</tr>
<tr>
<td>TI Frame Format Error</td>
<td>TIFE</td>
<td>TIFEIE</td>
<td>Writing TIFREC to 1</td>
<td>spi_it</td>
</tr>
<tr>
<td>Mode Fault</td>
<td>MODF</td>
<td>MODFIE</td>
<td>Writing MODFC to 1</td>
<td>spi_it</td>
</tr>
<tr>
<td>End Of Transfer (full transfer sequence completed - based on TSIZE value)</td>
<td>EOTF</td>
<td>EOTIE</td>
<td>Writing EOTC to 1</td>
<td>spi_it</td>
</tr>
<tr>
<td>Master mode suspended</td>
<td>SUSP</td>
<td>EOTIE</td>
<td>Writing SUSPC to 1</td>
<td>spi_it</td>
</tr>
<tr>
<td>TxFIFO transmission complete (TxFIFO empty)</td>
<td>TXCF</td>
<td>TXCIE</td>
<td>TXC cleared by HW when a transmission activity starts on the bus</td>
<td>spi_it</td>
</tr>
<tr>
<td>TSER value transferred to TSIZE (new value may be loaded to TSER)</td>
<td>TSERF</td>
<td>TSERFIE</td>
<td>Writing TSERFC to 1</td>
<td>spi_it</td>
</tr>
</tbody>
</table>

1. Refer to SPI2S register description for more details about the event flags.
55.8  **I2S main features**

- Full duplex communication
- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler
- Data length may be 16, 24 or 32 bits
- Channel length can be 16 or 32 in master, any value in slave
- Programmable clock polarity
- Error flags signaling for improved reliability: Underrun, Overrun and Frame Error
- Embedded Rx and TxFIFOs
- Supported I2S protocols:
  - I2S Philips standard
  - MSB-Justified standard (Left-Justified)
  - LSB-Justified standard (Right-Justified)
  - PCM standard (with short and long frame synchronization)
- Data ordering programmable (LSb or MSb first)
- DMA capability for transmission and reception
- Master clock can be output to drive an external audio component. The ratio is fixed at 256 x F_{WS} (where F_{WS} is the audio sampling frequency)
55.9  I2S functional description

55.9.1  I2S general description

The block diagram shown on Figure 577 also applies for I2S mode.

The SPI/I2S block can work on I2S/PCM mode, when the bit I2SMOD is set to 1. A dedicated register (SPI_I2SCFGR) is available for configuring the dedicated I2S parameters, which include the clock generator, and the serial link interface.

The I2S/PCM function uses the clock generator to produce the communication clock when the SPI/I2S is set in master mode. This clock generator is also the source of the master clock output (MCK).

Resources such as RxFIFO, TxFIFO, DMA and parts of interrupt signaling are shared with SPI function. The low-power mode function is also available in I2S mode, refer to Section 55.6: Low-power mode management and Section 55.10: I2S wakeup and interrupts.

55.9.2  Pin sharing with SPI function

The I2S shares four common pins with the SPI:

- SDO: Serial Data Output (mapped on the MOSI pin) to transmit the audio samples in master, and to receive the audio sample in slave. Refer to Section : Serial Data Line swapping on page 2133.

- SDI: Serial Data Input (mapped on the MISO pin) to receive the audio samples in master, and to transmit the audio sample in slave. Refer to Section : Serial Data Line swapping on page 2133.

- WS: Word Select (mapped on the SS pin) is the frame synchronization. It is configured as output in master mode, and as input for slave mode.

- CK: Serial Clock (mapped on the SCK pin) is the serial bit clock. It is configured as output in master mode, and as input for slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: Master Clock (mapped separately) is used, when the I2S is configured in master mode. The master clock rate is fixed to 256 x F_WS, where F_WS is the audio sampling frequency.
55.9.3 Bits and fields usable in I2S/PCM mode

When the I2S/PCM mode is selected (I2SMOD = ’1’), some bit fields are no longer relevant, and must be forced to a specific value in order to guarantee the behavior of the I2S/PCM function. Table 396 shows the list of bits and fields available in the I2S/PCM mode, and indicates which must be forced to a specific value.

Table 396. Bit fields usable in PCM/I2S mode

<table>
<thead>
<tr>
<th>Register name</th>
<th>Bit fields usable in PCM/I2S Mode</th>
<th>Constraints on other bit fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI/I2S control register 1 (SPI2S_CR1)</td>
<td>IOLOCK, CSUSP, CSTART</td>
<td>Other fields set to their reset values</td>
</tr>
<tr>
<td>SPI control register 2 (SPI.CR2)</td>
<td></td>
<td>Set to reset value</td>
</tr>
<tr>
<td>SPI configuration register 1 (SPI_CFG1)</td>
<td>TXDMAEN, RXDMAEN, FTHLV</td>
<td>Other fields set to their reset values</td>
</tr>
<tr>
<td>SPI configuration register 2 (SPI_CFG2)</td>
<td>AFCNTR, LBFRST, IOSWP</td>
<td>Other fields set to their reset values</td>
</tr>
<tr>
<td>SPI/I2S interrupt enable register (SPI2S_IER)</td>
<td>TIFREIE, OVRIE, UDRIE, TXPIE, RXPIE</td>
<td></td>
</tr>
<tr>
<td>SPI/I2S status register (SPI2S_SR)</td>
<td>RXWNE, RXPLVL, SUSP, TIFRE, OVR, UDR, TXP, RXP</td>
<td>Other flags not relevant</td>
</tr>
<tr>
<td>SPI/I2S interrupt/status flags clear register (SPI2S_IFCR)</td>
<td>SUSPC, TIFREC, OVRC, UDRC</td>
<td>Other fields set to their reset values</td>
</tr>
<tr>
<td>SPI/I2S transmit data register (SPI2S_TXDR)</td>
<td>The complete register</td>
<td>-</td>
</tr>
<tr>
<td>SPI/I2S receive data register (SPI2S_RXDR)</td>
<td>The complete register</td>
<td>-</td>
</tr>
<tr>
<td>SPI polynomial register (SPI_CRCPOLY)</td>
<td>-</td>
<td>Set to reset value</td>
</tr>
<tr>
<td>SPI transmitter CRC register (SPI_TXCRC)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPI receiver CRC register (SPI_RXCRC)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPI underrun data register (SPI_UDRDR)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPI/I2S configuration register (SPI_I2SCFGR)</td>
<td>The complete register</td>
<td>-</td>
</tr>
</tbody>
</table>
55.9.4 Slave and master modes

The SPI/I2S block supports master and slave mode for both I2S and PCM protocols. In master mode, both CK, WS and MCK signals are set to output.

In slave mode, both CK and WS signals are set to input. The signal MCK is not used in slave mode.

In order to improve the robustness of the SPI/I2S block in slave mode, the peripheral re-synchronizes each reception and transmission on WS signal. This means that:

- In I2S Philips standard, the shift-in or shift-out of each data is triggered one bit clock after each transition of WS.
- In I2S MSB justified standard, the shift-in or shift-out of each data is triggered as soon as a transition of WS is detected.
- In PCM standard, the shift-in or shift-out of each data is triggered one bit clock after the rising edge WS.

Note: This re-synchronization mechanism is not available for the I2S LSB justified standard.

Note: Note as well that there is no need to provide a kernel clock when the SPI/I2S is configured in slave mode.

55.9.5 Supported audio protocols

The I2S/PCM interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPIx_I2SCFGR register.

In the I2S protocol, the audio data are time-multiplexed on two channels: the left channel and the right channel. The WS signal is used to indicate which channel shall be considered as the left, and which one is the right.

In I2S master mode, four frames formats are supported:
- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel
- 32-bit data packed in a 32-bit channel

In PCM master mode, three frames formats are supported:
- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel
The figure hereafter shows the main definition used in this section: data length, channel length and frame length.

Figure 592. Waveform examples

- **MCK**: Sampling rate \( F_{WS} = \frac{F_{MCK}}{256} \)
- **WS**
- **CK** (CKPOL = 0)
- **SD** (In or Out)
- **Data length**
- **Channel length**
- **Frame length**

**I2S**

**PCM**

- **Sampling rate** \( F_{WS} = \frac{F_{MCK}}{256} \)
- **Channel length = Frame length**
I²S Philips standard

The I²S Philips standard is selected by setting I²SSTD to 0b00. This standard is supported in master and slave mode.

In this standard, the WS signal toggles one CK clock cycle before the first bit (MSb in I²S Philips standard) is available. A falling edge transition of WS indicates that the next data transferred is the left channel, and a rising edge transition indicates that the next data transferred is the right channel.

Figure 593. Master I²S Philips protocol waveforms (16/32-bit full accuracy)

CKPOL is set to 0 in order to match the I²S Philips protocol. See Selection of the CK sampling edge for information concerning the handling of WS signal.

Figure 593 shows an example of waveform generated by the SPI/I²S in the case where the channel length is equal to the data length. More precisely, this is true when CHLEN = 0 and DATLEN = 0b00 or when CHLEN = 1 and DATLEN = 0b10.

See Control of the WS Inversion for information concerning the handling of WS signal.

Figure 594. I²S Philips standard waveforms

In the case where the channel length is bigger than the data length, the remaining bits are forced to zero when the SPI/I²S is configured in transmit mode. This is applicable for both master and slave mode.
**MSB justified standard**

For this standard, the WS signal toggles when the first data bit is provided. The data transferred represents the left channel if WS is high, and the right channel if WS is low.

*Figure 595. Master MSB Justified 16-bit or 32-bit full-accuracy length*

CKPOL is set to 0 in order to match the I2S MSB justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of WS signal.

See *Control of the WS Inversion* for information concerning the handling of WS signal.

*Figure 596. Master MSB justified 16 or 24-bit data length*

In the case where the channel length is bigger than the data length, the remaining bits are forced to zero when the SPI/I2S is configured in master transmit mode. In slave transmit the remaining bits are forced to the value of the first bit of the next data to be generated in order to avoid timing issues (see *Figure 597*).
LSB justified standard

This standard is similar to the MSB justified standard in master mode (no difference for the 16 and 32-bit full-accuracy frame formats). The LSB justified 16 or 32-bit full-accuracy format give similar waveforms than MSB justified mode (see Figure 595) because the channel and data have the same length.

Note: In the LSB justified format, only 16 and 32-bit channel length are supported in master and slave mode. This is due to the fact that it is not possible to transfer properly the data if the channel length is not known by transmitter and receiver side.

Figure 598. LSB justified 16 or 24-bit data length

CKPOL is set to 0 in order to match the I2S LSB justified protocol. See Selection of the CK sampling edge for information concerning the handling of WS signal.

See Control of the WS Inversion for information concerning the handling of WS signal.
**PCM standard**

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and configurable using the PCMSYNC bit in SPI_I2SCFGR register.

*Note:* The difference between the PCM long and short frame, is just the width of the frame synchronization: for both protocols, the active edge of the frame is generated (or is expected for the Slave mode) one CK clock cycle before the first bit.

![Figure 599. Master PCM when the frame length is equal the data length](image)

For long frame synchronization, the WS signal assertion time is fixed to 13 bits in master mode.

A data size of 16 or 24 bits can be used when the channel length is set to 32 bits.

For short frame synchronization, the WS synchronization signal is only one cycle long.

See *Control of the WS Inversion* for information concerning the handling of WS signal.

![Figure 600. Master PCM standard waveforms (16 or 24-bit data length)](image)

If the PCM protocol is used in slave mode, frame lengths can be different from 16 or 32 bits. As shown in *Figure 601*, in slave mode various pulse widths of WS can be accepted as the start of frame is detected by a rising edge of WS. The only constraint is that the WS must go back to its inactive state for at least one CK cycle.
CKPOL is set to 0 in order to match the PCM protocol. See Selection of the CK sampling edge for information concerning the handling of WS signal.

55.9.6 Additional Serial Interface Flexibility

Variable frame length in slave

In slave mode, channel lengths different from 16 or 32 bits can be accepted, as long as the channel length is bigger than the data length. This is true for all protocols except for I2S LSB justified protocol.

Data ordering

For all data formats and communication standards, it is possible to select the data ordering (MSb or LSb first) thanks to the bit LSBFRST located into SPI configuration register 2 (SPI_CFG2).

Selection of the CK sampling edge

The CKPOL bit located into SPI/I2S configuration register (SPI_I2SCFGR) allows the user to choose the sampling edge polarity of the CK for slave and master modes, for all protocols.

- When CKPOL = 0, serial data SDO and WS (when master) are changed on the falling edge of CK and the serial data SDI and WS (when slave) are read on the rising edge.
- When CKPOL = 1, serial data SDO and WS (when master) are changed on the rising edge of CK and the serial data SDI and WS (when slave) are read on the falling edge.

Control of the WS Inversion

It is possible to invert the default WS signal polarity for master and slave modes, for all protocols, by setting WSINV to 1. By default the WS polarity is the following:

- In I2S Philips Standard, WS is low for left channel, and high for right channel
- In MSB/LSB justified mode, WS is high for left channel, and low for right channel
- In PCM mode, the start of frame is indicated by a rising edge of WS.
When WSINV is set to 1, the WS polarity is inverted, then:

- In I2S Philips Standard, WS is high for left channel, and low for right channel
- In MSB/LSB justified mode, WS is low for left channel, and high for right channel
- In PCM mode, the start of frame is indicated by a falling edge of WS.

WSINV is located into SPI/I2S configuration register (SPI_I2SCFGR).

Control of the IOs

The SPI/I2S block allows the settling of the WS and CK signals to their inactive state before enabling the SPI/I2S thanks to the AFCNTR bit of SPI configuration register 2 (SPI_CFG2).

This can be done by programming CKPOL and WSINV using the following sequence:

Assuming that AFCNTR is initially set to 0
- Set I2SMOD = 1, (In order to inform the hardware that the CK and WS polarity is controlled via CKPOL and WSINV).
- Set bits CKPOL and WSINV to the wanted value.
- Set AFCNTR = 1. Then the inactive level of CK and WS IOs is set according to CKPOL and WSINV values, even if the SPI/I2S is not yet enabled.
- Then performs the activation sequence of the I2S/PCM

Table 397 shows the level of WS and CK signals, when the AFCNTR bit is set to 1, and before the SPI/I2S block is enabled (i.e. inactive level). Note that the level of WS depends also on the protocol selected.

### Table 397. WS and CK level before SPI/I2S is enabled when AFCNTR = 1

<table>
<thead>
<tr>
<th>WSINV</th>
<th>I2SSSTD</th>
<th>WS level before SPI/I2S is enabled</th>
<th>CKPOL</th>
<th>CK level before SPI/I2S is enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>I2S Std (00)</td>
<td>High</td>
<td>0</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Others</td>
<td>Low</td>
<td>1</td>
<td>High</td>
</tr>
<tr>
<td>1</td>
<td>I2S Std (00)</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Others</td>
<td>High</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The bit AFCNTR shall not be set to 1, when the SPI/I2S is in slave mode.

Serial Data Line swapping

The SPI/I2S offers the possibility to swap the function of SDI and SDO lines thanks to IOSWP bit located into SPI configuration register 2 (SPI_CFG2). Table 398 gives details on this feature.

### Table 398. Serial data line swapping

<table>
<thead>
<tr>
<th>Configuration</th>
<th>IOSWP</th>
<th>SDI direction</th>
<th>SDO direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master/slave RX</td>
<td>0</td>
<td>IN</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-</td>
<td>IN</td>
</tr>
</tbody>
</table>
55.9.7 Start-up sequence

When the bit SPE is set to 0, the user is not allowed to read and write into the SPI2S_RXDR and SPI2S_TXDR registers, but the access to other registers is allowed.

When the application wants to use the SPI/I2S block the user has to proceed as follow:

1. Insure that the SPE is set to 0, otherwise write SPE to 0.
2. Program all the configuration and control registers according to the wanted configuration. Refer to Section 55.9.16 for detailed programming examples.
3. Set the SPE bit to 1, in order to activate the SPI/I2S block. When this bit is set, the serial interface is still disabled, but the DMA and interrupt services are working, allowing for example, the data transfer into the TxFIFO.
4. Set bit CSTART to 1, in order to activate the serial interface.

As shown in Figure 602, in I2S Philips standard master TX, the generation of the WS, MCK and CK signals is started as soon as the bit CSTART is set to 1 and the TxFIFO is not empty. Note that the bit clock CK is activated 4 rising edges before the falling edge of WS in order to insure that the external slave device can detect properly WS transition. Other standards behave similarly.

Figure 602. Start-up sequence, I2S Philips standard, master

For simplification, the waveforms shown in the I2S functional description section have been done with IOSWP = 0.
Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after about 3 periods of CK clock (SYNC_DLY).

In slave mode, once the bit CSTART is set to 1, the data transfer starts when the start-of-frame condition is met:

- For I2S Philips standard, the start-of-frame condition is a falling edge of WS signal. The transmission/reception starts one bit clock later.
  
  If WSINV = 1, then the start-of-frame condition is a rising edge.

- For other protocols, the start-of-frame condition is a rising edge of WS signal. The transmission/reception starts at rising edge of WS for MSB aligned protocol. The transmission/reception starts one bit clock later for PCM protocol.
  
  If WSINV = 1, then the start-of-frame condition is a falling edge.

Figure 603 shows an example of start-up sequence in I2S Philips standard, slave mode.

Figure 603. Start-up sequence, I2S Philips standard, slave

Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after 2 periods of CK clock (SYNC_DLY).

55.9.8 Stop sequence

The application can stop the I2S/PCM transfers by setting the SPE bit to 0. In that case the communication is stopped immediately, without waiting for the end of the current frame.

In master mode it is also possible to stop the I2S/PCM transfers at the end of the current frame. For that purpose, the user has to set the bit CSUSP to 1, and polls the CSTART bit until it goes to 0. The CSTART bit goes to 0 when the current stereo (if an I2S mode was selected) or mono sample are completely shifted in or out. Then the SPE bit can be set to 0.

The Figure 604 shows an example of stop sequence in the case of master mode. The CSUSP bit is set to 1, during the transmission of left sample, the transfer continue until the last bit of the right sample is transferred. Then CSTART and CSUSP go back to 0, CK and WS signals go back to their inactive state, and the user can set SPE to 0.
55.9.9 Clock generator

When the I2S or PCM is configured in master mode, the user needs to program the clock generator in order to produce the Frame Synchronization (WS), the bit clock (CK) and the master clock (MCK) at the desired frequency.

If the I2S or PCM is used in slave mode, there is no need to configure the clock generator.

Note: In slave mode, the stop sequence is only controlled by the SPE bit.

The frequency generated on MCK, CK and WS depends mainly on I2SDIV, ODD, CHLEN and MCKOE. The bit MCKOE indicates if a master clock need to be generated or not. The master clock has a frequency 256 times higher than the frame synchronization. This master clock is often required to provide a reference clock to external audio codecs.

Note: In master mode, there is no specific constraints on the ratio between the bus clock rate ($F_{pclk}$) and the bit clock ($F_{CK}$). The bus clock frequency must be high enough in order to support the data throughput.
When the master clock is generated (MCKOE = 1), the frequency of the frame synchronization is given by the following formula in I2S mode:

\[
F_{WS} = \frac{F_{i2s\_clk}}{256 \times \{(2 \times I2SDIV) + ODD\}}
\]

and by this formula in PCM mode:

\[
F_{WS} = \frac{F_{i2s\_clk}}{128 \times \{(2 \times I2SDIV) + ODD\}}
\]

In addition, the frequency of the MCK (\(F_{MCK}\)) is given by the formula:

\[
F_{MCK} = \frac{F_{i2s\_clk}}{\{(2 \times I2SDIV) + ODD\}}
\]

When the master clock is disabled (MCKOE = 0), the frequency of the frame synchronization is given by the following formula in I2S mode:

\[
F_{WS} = \frac{F_{i2s\_clk}}{32 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}
\]

And by this formula in PCM mode:

\[
F_{WS} = \frac{F_{i2s\_clk}}{16 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}
\]

Where \(F_{WS}\) is the frequency of the frame synchronization, and \(F_{i2s\_clk}\) is the frequency of the kernel clock provided to the SPI/I2S block.

**Note:**
- \(CHLEN\) and \(ODD\) can be either 0 or 1.
- \(I2SDIV\) can take any values from 0 to 255 when \(ODD = 0\), but when \(ODD = 1\), the value \(I2SDIV = 1\) is not allowed.
- When \(I2SDIV = 0\), then \({(2 \times I2SDIV) + ODD}\) is forced to 1.

**Note:**
- When \({(2 \times I2SDIV) + ODD}\) is odd, the duty cycle of MCK or the CK signals is not 50%.
- Care must be taken when odd ratio is used: it can impact margin on setup and hold time.
- For example if \({(2 \times I2SDIV) + ODD}\) = 5, then the duty cycle can be 40%.

Table 399 provides examples of clock generator programming for I2S modes.

**MCK Generation**

The master clock MCK can be generated regardless to the SPE bit. The MCK generating is controlled by the following bits:

- I2SMOD must equal to 1,
- I2SCFG must select a master mode,
- MCKOE must be set to 1
55.9.10 Internal FIFOs

The I2S interface can use a dedicated FIFO for the RX and the TX path. The samples to transmit can be written into the TxFIFO via the SPI2S_TXDR register. The reading of RxFIFO is performed via the SPI2S_RXDR register.

Data alignment and ordering

It is possible to select the data alignment into the SPI2S_RXDR and SPI2S_TXDR registers thanks to the DATFMT bit.

Note as well that the format of the data located into the SPI2S_RXDR or SPI2S_TXDR depends as well on the way those registers are accessed via the APB bus.

Figure 606 shows the allowed settings between APB access sizes, DATFMT and DATLEN.

Note: Caution shall be taken when the APB access size is 32 bits, and DATLEN = 0. For read operation the RxFIFO must contain at least two data, otherwise the read data are invalid. In the same way, for write operation, the TxFIFO must have at least two empty locations, otherwise a data can be lost.

<table>
<thead>
<tr>
<th>i2s clk (MHz)</th>
<th>Channel length (bits)</th>
<th>I2SDIV</th>
<th>ODD</th>
<th>MCK</th>
<th>Sampling rate: FWS (kHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.288</td>
<td>16</td>
<td>12</td>
<td>0</td>
<td>No</td>
<td>16</td>
</tr>
<tr>
<td>12.288</td>
<td>32</td>
<td>6</td>
<td>0</td>
<td>No</td>
<td>16</td>
</tr>
<tr>
<td>12.288</td>
<td>16</td>
<td>6</td>
<td>0</td>
<td>No</td>
<td>32</td>
</tr>
<tr>
<td>12.288</td>
<td>32</td>
<td>3</td>
<td>0</td>
<td>No</td>
<td>32</td>
</tr>
<tr>
<td>49.152</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>No</td>
<td>48</td>
</tr>
<tr>
<td>49.152</td>
<td>32</td>
<td>8</td>
<td>0</td>
<td>No</td>
<td>48</td>
</tr>
<tr>
<td>49.152</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td>No</td>
<td>96</td>
</tr>
<tr>
<td>49.152</td>
<td>32</td>
<td>4</td>
<td>0</td>
<td>No</td>
<td>96</td>
</tr>
<tr>
<td>49.152</td>
<td>16</td>
<td>4</td>
<td>0</td>
<td>No</td>
<td>192</td>
</tr>
<tr>
<td>49.152</td>
<td>32</td>
<td>2</td>
<td>0</td>
<td>No</td>
<td>192</td>
</tr>
<tr>
<td>4.096</td>
<td>16 or 32</td>
<td>0</td>
<td>-</td>
<td>Yes</td>
<td>16</td>
</tr>
<tr>
<td>24.576</td>
<td>16 or 32</td>
<td>3</td>
<td>0</td>
<td>Yes</td>
<td>32</td>
</tr>
<tr>
<td>49.152</td>
<td>16 or 32</td>
<td>3</td>
<td>0</td>
<td>Yes</td>
<td>48</td>
</tr>
<tr>
<td>12.288</td>
<td>16 or 32</td>
<td>0</td>
<td>-</td>
<td>Yes</td>
<td>48</td>
</tr>
<tr>
<td>49.152</td>
<td>16 or 32</td>
<td>2</td>
<td>0</td>
<td>Yes</td>
<td>96</td>
</tr>
<tr>
<td>61.44</td>
<td>16 or 32</td>
<td>2</td>
<td>1</td>
<td>Yes</td>
<td>192</td>
</tr>
<tr>
<td>98.304</td>
<td>16 or 32</td>
<td>2</td>
<td>0</td>
<td>Yes</td>
<td>192</td>
</tr>
<tr>
<td>196.608</td>
<td>16 or 32</td>
<td>2</td>
<td>0</td>
<td>Yes</td>
<td>192</td>
</tr>
</tbody>
</table>
1. In I2S mode, the sample N represents the left sample, and the sample N+1 is the right sample.

It is possible to generate an interrupt or a DMA request according to a programmable FIFO threshold levels. The FIFO threshold is common to RX and TxFIFOs can be adjusted via FTHLV.

In I2S mode, the left and right audio samples are interleaved into the FIFOs. It means that for transmit operations, the user has to start to fill-up the TxFIFO with a left sample, followed by a right sample, and so on. For receive mode, the first data read from the RxFIFO is supposed to represent a left channel, the next one is a right channel, and so on.

Note that the read and write pointers of the FIFOs are reset when the bit SPE is set to 0.

Refer to Section 55.9.11 and Section 55.9.15 for additional information.

FIFO size optimization

The basic element of the FIFO is the byte. This allows an optimization of the FIFO locations. For example when the data size is fixed to 24 bits, each audio sample takes 3 basic FIFO elements.

For example, a FIFO with 16 basic elements can have a depth of:
- 8 samples, if the DATLEN = 0 (16 bits),
- 5 samples, if the DATLEN = 1 (24 bits),
- 4 samples, if the DATLEN = 2 (32 bits).

55.9.11 FIFOs status flags

Two status flags are provided for the application to fully monitor the state of the I2S interface. Both flags can generate an interrupt request. The receive interrupt is generated if RXPIE bit is enabled, the transmit interrupt is generated if TXPIE bit is enabled. Those bits are located into the SPI_IER register.

TxFIFO threshold reached (TXP)

When set, this flag indicates that the TxFIFO contains at least FTHLV empty locations. Thus FTHLV new data to be transmitted can be written into SPI2S_TXDR. The TXP flag is reset when the amount of empty locations is lower than FTHLV. Note that TXP = 1, when the I2S is disabled (SPE bit is reset).
RxFIFO threshold reached (RXP)

When set, this flag indicates that there is at least FTHLV valid data into the Rx FIFO, thus the user can read those data via SPI2S_RXDR. It is reset when the Rx FIFO contains less than FTHLV data.

See Section 55.10 for additional information on interrupt function in I2S mode.

55.9.12 Handling of underrun situation

In transmit mode, the UDR flag is set when a new data needs to be loaded into the shift register while the Tx FIFO is already empty. In such a situation at least a data is lost.

In I2S mode, there is a hardware mechanism in order to prevent misalignment situation (left and right channel swapped). As shown in the following figure, when an underrun occurs, the peripheral re-plays the last valid data on left and right channels as long as conditions of restart are not met. The transmission restarts:

- When there is enough data into the Tx FIFO, and
- When the UDR flag is cleared by the software,

Then the next data transmitted is:

- A right channel if the underrun occurred when a right channel data needed to be transmitted, or
- A left channel if the underrun occurred when a left channel data needed to be transmitted.

The UDR flag can trigger an interrupt if the UDRIE bit in the SPI_IER register is set. The UDR bit is cleared by writing UDRC bit of SPI_IFCR register to 1.

When the block is configured in PCM mode, this re-alignment mechanism is not activated.

Note: An underrun situation can occur in master or slave mode. In master mode, when an underrun occurs, the WS, CK and MCK signal are not gated.

Due to resynchronization, any change on the UDR flag is taken into account by the hardware after at least 2 periods of CK clock.
55.9.13 Handling of overrun situation

The OVR flag is set when received data need to written into the RxFIFO, while the RxFIFO is already full. As a result, some incoming data are lost.

In I2S mode, there is a hardware mechanism in order to prevent misalignment situation (left and right channel swapped). As shown in the following figure, when an overrun occurs, the peripheral stops writing data into the RxFIFO as long as conditions of restart are not met. When there is enough room into the RxFIFO, and the OVR flag is cleared, the block starts by writing next the right channel into the RxFIFO if the overrun occurred when a right channel data was received or by writing the next left channel if the overrun occurred when a left channel data was received.

An interrupt may be generated if the OVRIE bit is set in the SPI_IER register. The OVR bit is cleared by writing OVRC bit of SPI_IFCR register to 1.

When the block is configured in PCM mode, this re-alignment mechanism is not activated

Note: An overrun situation can occur in master or slave mode. In master mode, when an overrun occurs, the WS, CK and MCK signal are not gated.

55.9.14 Frame error detection

When configured in slave mode, the SPI/I2S block detects two kinds of frame errors:

- A frame synchronization received while the shift-in or shift-out of the previous data is not completed (early frame error). This mode is selected with FIXCH = 0.
- A frame synchronization occurring at an unexpected position. This mode is selected with FIXCH = 1.

In slave mode, if the frame length provided by the external master device is different from 32 or 64 bits, the user has to set FIXCH to 0. As the SPI/I2S synchronize each transfer with the WS there is no misalignment risk, but in a noisy environment, if a glitch occurs in the CK signal, a sample may be affected and the application is not aware of this.

If the frame length provided by the external master device is equal to 32 or 64 bits, then the user can set FIXCH to 1 and adjust accordingly CHLEN. As the SPI/I2S synchronize each transfer with the WS there is still no misalignment risk, and if the amount of bit clock
between each channel boundary is different from CHLEN, the frame error flag (TIFRE) is set to 1.

*Figure 609* shows an example of frame error detection. The SPI/I2S block is in slave mode and the amount of bit clock periods for left channel are not enough to shift-in or shift-out the data. The figure shows that the on-going transfer is interrupted and the next one is started in order to remain aligned to the WS signal.

*Figure 609. Frame error detection, with FIXCH=0*

An interrupt can be generated if the TIFREIE bit is set. The frame error flag (TIFRE) is cleared by writing the TIFREC bit of the SPI_IFCR register to 1.

It is possible to extend the coverage of the frame error flag by setting the bit FIXCH to 1. When this bit is set to 1, then the SPI/I2S is expecting fixed channel lengths in slave mode. This means that the expected channel length can be 16 or 32 bits, according to CHLEN. As shown in *Figure 610*, in this mode the SPI/I2S block is able to detect if the WS signal is changing at the expected moment (too early or too late).

*Note:* *Figure 609* and *Figure 610* show the mechanism for the slave transmit mode, but this is also true for slave receive and slave full-duplex.

*Figure 610. Frame error detection, with FIXCH=1*

The frame error detection can be generally due to noisy environment disturbing the good reception of WS or CK signals.

*Note:* The SPI/I2S is not able to recover properly if an overrun and an early frame occur within the same frame. In this case the user has to disable and re-enable the SPI/I2S.
### 55.9.15 DMA Interface

The I2S/PCM mode shares the same DMA requests lines than the SPI function. There is a separated DMA channel for TX and RX paths. Each DMA channel can be enabled via RXDMAEN and TXDMAEN bits of SPI_CFG1 register.

In receive mode, the DMA interface is working as follow:

1. The hardware evaluates the RxFIFO level,
2. If the RxFIFO contains at least FTHLV samples, then FTHLV DMA requests are generated,
   - When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the RxFIFO contains less than FTHLV samples, no DMA request is generated, and the hardware loop to step 1

In transmit mode, the DMA interface is working as follow:

1. The hardware evaluates the TxFIFO level,
2. If the TxFIFO contains at least FTHLV empty locations, then FTHLV DMA requests are generated,
   - When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the TxFIFO contains less than FTHLV empty locations, no DMA request is generated, and the hardware loop to step 1
55.9.16 Programing examples

Master I2S Philips standard, transmit

This example shows how to program the interface for supporting the Philips I2S standard in master transmit mode, with a sampling rate of 48 kHz, using the master clock. The assumption taken is that SPI/I2S is receiving a kernel clock (i2s_clk) of 61.44 MHz from the clock controller of the circuit.

Start Procedure

1. Enable the bus interface clock (pclk or hclk), release the reset signal if needed in order to be able to program the SPI/I2S block.
2. Insure that the SPI/I2S block receives properly a kernel frequency (at 61.44 MHz in this example).
3. Insure that SPE is set to 0.
4. Program the clock generator in order to provide the MCK clock and to have a frame synchronization rate at exactly 48 kHz. So I2SDIV = 2, ODD = 1, and MCKOE = 1.
5. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 0 (Philips Standard), I2SCFG = 2 (master transmit), I2SMOD = 1, for I2S/PCM mode. The register SPI_I2SCFGR must be updated before going to next steps.
6. Adjust the FIFO threshold, by setting the wanted value into FTHLV. For example if a threshold of 2 audio samples is required, FTHLV = 1.
7. Clear all status flag registers.
8. Enable the flags who shall generate an interrupt such as UDRIE. Note that TIFRE is not meaningful in master mode.
9. If the data transfer uses DMA:
   − Program the DMA peripheral,
   − Initialize the memory buffer with valid audio samples,
   − Enable the DMA channel,
10. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the TXPIE bit to 1.
11. Set SPE to 1, as soon as this bit is set to one the following actions may happen:
    − If the interrupt generation is enabled, the SPI/I2S generates an interrupt request allowing the interrupt handler to fill-up the TxFIFO.
    − If the DMA transfer are enabled (TXDMAEN = 1), the SPI/I2S generates DMA requests in order to fill-up the TxFIFO
12. Finally, the user has to insure that the TxFIFO is not empty before enabling the serial interface. This is important in order to avoid an underrun condition when the interface is enabled. Then the SPI/I2S block can be enabled by setting the bit CSTART to 1. CSTART bit is located into SPI_CR1 register.

Stop Procedure in master mode

1. Set the bit CSUSP to 1, in order to stop on-going transfers
2. Check the value of CSTART bit until it goes to 0
3. Stop DMA peripheral, bus clock...
4. Set bit SPE to 0 in order to disable the SPI/I2S block
Master I2S MSB Aligned, full-duplex

This example shows how to program the interface for supporting the I2S MSB aligned protocol in master full-duplex mode, with a sampling rate of 48 kHz, without using the master clock. We took the assumption that the SPI/I2S is receiving a kernel clock (i2s_clk) of 12.288 MHz from the clock controller of the circuit.

Procedure

1. Enable the bus interface clock (pclk or hclk), release the reset signal if needed in order to be able to program the SPI/I2S block.
2. Insure that the SPI/I2S block receives properly a kernel frequency (at 12.288 MHz in this example).
3. Insure that SPE is set to 0.
4. Program the clock generator in order to provide the MCK clock, and to have a frame synchronization rate at exactly 48 kHz. So I2SDIV = 2, ODD = 0, and MCKOE = 0.
5. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 1 (MSB Justified), I2SCFG = 5 (master Full-duplex), I2SMOD = 1, for I2S/PCM mode. The register SPI_I2SCFGR must be updated before going to next steps.
6. Adjust the FIFO threshold, by setting the wanted value into FTHLV. For example if a threshold of 2 audio samples is required, FTHLV = 1.
7. Clear all status flag registers.
8. Enable the flags who shall generate an interrupt such as UDRIE. Note that TIFRE is not meaningful in master mode.
9. If the data transfer uses DMA:
   − Program the DMA peripheral: two channels, one for RX and one for TX
   − Initialize the memory buffer with valid audio samples for TX path
   − Enable the DMA channels,
   − In the SPI/I2S block, enable the DMA by setting the TXDMAEN and RXDMAEN bits to 1. As soon as these bits are set to 1, the SPI/I2S start to fill-up the TxFIFO by sending DMA requests
10. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the TXPIE and RXPIE bits to 1.
11. Set SPE to 1, as soon as this bit is set to one the following actions may happen:
    − If the interrupt generation is enabled, the SPI/I2S generates an interrupt request allowing the interrupt handler to fill-up the TxFIFO.
    − If the DMA transfer are enabled, the SPI/I2S generates DMA requests in order to fill-up the TxFIFO
12. Finally, the user has to insure that the TxFIFO is not empty before enabling the serial interface. This is important in order to avoid an underrun condition when the interface is enabled. Then the SPI/I2S block can be enabled by setting the bit CSTART to 1. CSTART bit is located into SPI_CR1 register.

Refer to Stop Procedure in master mode for details on the stop sequence.
55.9.17 Slave I2S Philips standard, receive

This example shows how to program the interface for supporting the I2S Philips standard protocol in slave receiver mode, with a sampling rate of 48 kHz. Note that in slave mode the SPI/I2S block cannot control the sample rate of the received samples. In this example we took the assumption that the external master device is delivering an I2S frame structure with a channel length of 24 bits. So we cannot use the capability offered for frame error detection when FIXCH is set to 1.

Procedure

1. Enable the bus interface clock (pclk or hclk), release the reset signal if needed in order to be able to program the SPI/I2S block.
2. Insure that SPE is set to 0.
3. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, FIXCH = 0 (because channel length is different from 16 and 32 bits), DATLEN = 0 (16 bits), I2SSTD = 0 (Philips protocol), I2SCFG = 1 (slave RX), I2SMOD = 1, for I2S mode. The register SPI_I2SCFGR must be properly programmed before going to next steps.
4. Adjust the FIFO threshold, by setting the wanted value into FTHLV. For example if a threshold of 2 audio samples is required, FTHLV = 1.
5. Clear all status flag registers.
6. Enable the flags who shall generate an interrupt such as UDRIE and TIFRE.
7. If the data transfer uses DMA:
   – Program the DMA peripheral: one RX channel
   – Enable the DMA channel,
   – In the SPI/I2S block, enable the DMA by setting the RXDMAEN bit to 1.
8. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the RXPIE bit to 1.
9. Set SPE to 1.
10. Finally the user can set the bit CSTART to 1 in order to enable the serial interface. The SPI/I2S starts to store data into the RxFIFO on the next occurrence of left data transmitted by the external master device.

Stop Procedure in slave mode

1. Set bit SPE to 0 in order to disable the SPI/I2S block
2. Stop DMA peripheral, bus clock...
55.10 I2S wakeup and interrupts

In PCM/I2S mode an interrupt (spi_it) or a wakeup event signal (spi_wkup) can be generated according to the events described in the Table 400.

Interrupt events can be enabled and disabled separately.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Event clear method</th>
<th>Interrupt/Wake up activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxFIFO threshold reached</td>
<td>TXP</td>
<td>TXPIE</td>
<td>TXP flag is cleared when the TxFIFO contains less than FTHLV empty locations</td>
<td>YES</td>
</tr>
<tr>
<td>RxFIFO threshold reached</td>
<td>RXP</td>
<td>RXPIE</td>
<td>RXP flag is cleared when the RxFIFO contains less than FTHLV samples</td>
<td>YES</td>
</tr>
<tr>
<td>Overrun error</td>
<td>OVR</td>
<td>OVRIE</td>
<td>OVR is cleared by writing OVRC to 1</td>
<td>YES</td>
</tr>
<tr>
<td>Underrun error</td>
<td>UDR</td>
<td>UDRIE</td>
<td>UDR is cleared by writing UDRC to 1</td>
<td>NO</td>
</tr>
<tr>
<td>Frame error flag</td>
<td>TIFRE</td>
<td>TIFREIE</td>
<td>TIFRE is cleared by writing TIFREC to 1</td>
<td>NO</td>
</tr>
</tbody>
</table>
55.11 SPI/I2S registers

55.11.1 SPI/I2S control register 1 (SPI2S_CR1)

Address offset: 0x00
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>w</td>
<td>rs</td>
<td>nw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 16 IOLOCK: locking the AF configuration of associated I/Os
This bit is set by software and cleared by hardware on next device reset
0: AF configuration is not locked
1: AF configuration is locked
When this bit is set, SPI_CFG2 register content cannot be modified any more.
This bit must be configured when the SPI is disabled. When SPE=1, it is write protected.

Bit 15 TCRCINI: CRC calculation initialization pattern control for transmitter
0: All zero pattern is applied
1: All ones pattern is applied

Bit 14 RCRCINI: CRC calculation initialization pattern control for receiver
0: all zero pattern is applied
1: all ones pattern is applied

Bit 13 CRC33_17: 32-bit CRC polynomial configuration
0: full size (33-bit or 17-bit) CRC polynomial is not used
1: full size (33-bit or 17-bit) CRC polynomial is used

Bit 12 SSI: internal SS signal input level
This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the peripheral SS input and the I/O value of the SS pin is ignored.

Bit 11 HDDIR: Rx/Tx direction at Half-duplex mode
In Half-Duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data transfer. This bit is ignored in Full-Duplex or any Simplex configuration.
0: SPI is Receiver
1: SPI is transmitter

Bit 10 CSUSP: master suspend request
This bit reads as zero.
In master mode, when this bit is set by software, CSTART bit is reset at the end of the current frame and SPI communication is suspended. The user has to check SUSP flag to check end of the frame transaction.
The master mode communication must be suspended (using this bit or keeping TXDR empty) before disabling the SPI or going to low-power mode.
Bit 9 **CSTART**: master transfer start
This bit is set by software to start an SPI or I2S/PCM communication. In SPI mode, it is cleared by hardware when End Of Transfer (EOT) flag is set or when a transaction suspend request is accepted. In I2S/PCM mode, it is also cleared by hardware as described in the section stop sequence.
0: master transfer is at idle
1: master transfer is on-going or temporary suspended by automatic suspend
In SPI mode, CSTART can be set only when SPE=1 and MASTER=1.
In SPI mode, in case of master transmission is enabled, communication starts or continues only if any data is available in the transmission FIFO.
In I2S/PCM mode, CSTART can be set when SPE = 1.

Bit 8 **MASRX**: master automatic SUSP in Receive mode
This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.
0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)
1: SPI flow is suspended temporary on RxFIFO full condition, before reaching overrun condition. The SUSP flag is set when SPI communication is suspended.
When SPI communication is suspended to prevent overrun condition it could happen that few bits of next frame are already clocked out due to internal synchronization delay. Once the RxFIFO is read, the communication resumes after software clearing of the SUSP bit and continues by subsequent bits transaction without any next constraint.
For the same reason, the automatic suspension is not quite reliable when size of data drops below 8 bits. In this case, a safe suspension can be achieved by combination with delay inserted between data frames applied when MIDI parameter keeps a nonzero value; sum of data size and the interleaved SPI cycles must always produce interval at length of 8 SPI clock periods at minimum.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SPE**: serial peripheral enable
This bit is set by and cleared by software.
0: serial peripheral disabled.
1: serial peripheral enabled
When SPE=1, SPI data transfer is enabled. Configuration registers and IOLOCK bit in SPI_CR1 are write protected. They can be changed only when SPE=0.
When SPI=0 any SPI operation is stopped and disabled, internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero.
SPE cannot be set when MODF error flag is active.

### 55.11.2 SPI control register 2 (SPI_CR2)
Address offset: 0x04
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>TSER[15:0]</td>
</tr>
<tr>
<td>14</td>
<td>rs</td>
</tr>
<tr>
<td>13</td>
<td>rs</td>
</tr>
<tr>
<td>12</td>
<td>rs</td>
</tr>
<tr>
<td>11</td>
<td>rs</td>
</tr>
<tr>
<td>10</td>
<td>rs</td>
</tr>
<tr>
<td>9</td>
<td>rs</td>
</tr>
<tr>
<td>8</td>
<td>rs</td>
</tr>
<tr>
<td>7</td>
<td>rs</td>
</tr>
<tr>
<td>6</td>
<td>rs</td>
</tr>
<tr>
<td>5</td>
<td>rs</td>
</tr>
<tr>
<td>4</td>
<td>rs</td>
</tr>
<tr>
<td>3</td>
<td>rs</td>
</tr>
<tr>
<td>2</td>
<td>rs</td>
</tr>
<tr>
<td>1</td>
<td>rs</td>
</tr>
<tr>
<td>0</td>
<td>rs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>TSIZE[15:0]</td>
</tr>
<tr>
<td>14</td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:16  **TSER[15:0]:** number of data transfer extension to be reload into TSIZE just when a previous number of data stored at TSIZE is transacted

This register can be set by software when its content is cleared. It is cleared by hardware once TSIZE reload is done. The TSER value must be programmed in advance before CTSIZE counter reaches zero otherwise the reload is not taken into account and traffic terminates with normal EOT event.

Bits 15:0  **TSIZE[15:0]:** number of data at current transfer

When these bits are changed by software, the SPI must be disabled. The field can be updated by hardware optionally, too, to be reloaded by the TSER value if applicable.

Endless transaction is initialized when CSTART is set while zero value is stored at TSIZE. TSIZE cannot be set to 0xFFFF value when CRC is enabled.

### 55.11.3 SPI configuration register 1 (SPI_CFG1)

Address offset: 0x08

Reset value: 0x0007 0007

**Content of this register is write protected when SPI is enabled**

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

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

Bit 31  Reserved, must be kept at reset value.

Bits 30:28  **MBR[2:0]:** master baud rate

000: SPI master clock/2
001: SPI master clock/4
010: SPI master clock/8
011: SPI master clock/16
100: SPI master clock/32
101: SPI master clock/64
110: SPI master clock/128
111: SPI master clock/256

Bits 27:23  Reserved, must be kept at reset value.

Bit 22  **CRCEN:** hardware CRC computation enable

0: CRC calculation disabled
1: CRC calculation Enabled

Bit 21  Reserved, must be kept at reset value.
Bits 20:16 CRCSIZE[4:0]: length of CRC frame to be transacted and compared
Most significant bits are taken into account from polynomial calculation when CRC result is transacted or compared. The length of the polynomial is not affected by this setting.
00000: reserved
00001: reserved
00010: reserved
00011: 4-bits
00100: 5-bits
00101: 6-bits
00110: 7-bits
00111: 8-bits
.....
11101: 30-bits
11110: 31-bits
11111: 32-bits
The value must be equal or multiply of DSIZE length
Note: If crc calculation is disabled by CRCEN=0, the CRCSIZE field must be kept at its default setting.
Note: The most significant bit at CRCSIZE bit field is reserved at the peripheral instances where the data size is limited to 16-bit.

Bit 15 TXDMAEN: Tx DMA stream enable
0: Tx DMA disabled
1: Tx DMA enabled

Bit 14 RXDMAEN: Rx DMA stream enable
0: Rx-DMA disabled
1: Rx-DMA enabled

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 UDRDET[1:0]: detection of underrun condition at slave transmitter
00: underrun is detected at begin of data frame (no protection of 1-st bit)
01: underrun is detected at end of last data frame
10: underrun is detected by begin of active SS signal
11: reserved
The user can define here when and how the underrun condition is detected at slave receiver
Bits 10:9  **UDRCFG[1:0]**: behavior of slave transmitter at underrun condition
- 00: slave sends a constant pattern defined by the user at SPI_UISR register
- 01: slave repeats lastly received data frame from master
- 10: slave repeats its lastly transmitted data frame
- 11: reserved

While TxFIFO is empty shift register for transmission keeps value of SPI_UISR register. This register can be either locked to send value defined by the user or refreshed automatically by lastly transacted frame received from master or updated by data stored lastly at slave TxFIFO (by write to TXDR).

Bits 8:5  **FTHLV[3:0]**: FIFO threshold level
Defines number of data frames at single data packet. The size of the packet must not exceed 1/2 of FIFO space.
- 0000: 1-data
- 0001: 2-data
- 0010: 3-data
- 0011: 4-data
- 0100: 5-data
- 0101: 6-data
- 0110: 7-data
- 0111: 8-data
- 1000: 9-data
- 1001: 10-data
- 1010: 11-data
- 1011: 12-data
- 1100: 13-data
- 1101: 14-data
- 1110: 15-data
- 1111: 16-data

SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:
- If SPI data register is accessed as a 16-bit register and DSIZE<=8bit, better to select FTHLV=2, 4, 6 etc,
- If SPI data register is accessed as a 32-bit register and DSIZE>8bit, better to select FTHLV=2, 4, 6 etc, while if DSIZE<=8bit, better to select FTHLV=4, 8, 12 etc

Bits 4:0  **DSIZE[4:0]**: number of bits in at single SPI data frame
- 00000: not used
- 00001: not used
- 00010: not used
- 00011: 4-bits
- 00100: 5-bits
- 00101: 6-bits
- 00110: 7-bits
- 00111: 8-bits
- ....
- 11001: 30-bits
- 11100: 31-bits
- 11101: 32-bits

*Note: The most significant bit at DSIZE bit field is reserved at the peripheral instances where data size is limited to 16-bit.*
55.11.4 SPI configuration register 2 (SPI_CFG2)

Address offset: 0x0C
Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled or IOLOCK bit is set at SPI2S_CR1 register.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>AFCNTR: alternate function GPIOs control</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is taken into account when SPE=0 only</td>
</tr>
<tr>
<td></td>
<td>0: the peripheral takes no control of GPIOs while it is disabled</td>
</tr>
<tr>
<td></td>
<td>1: the peripheral keeps always control of all associated GPIOs</td>
</tr>
<tr>
<td></td>
<td>When SPI master has to be disabled temporary for a specific configuration reason (e.g. CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding the current SPI configuration. This bit must be never used at slave mode as any slave transmitter must not force its MISO output once the SPI is disabled.</td>
</tr>
<tr>
<td>Note:</td>
<td>This bit can be also used in PCM and I2S modes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>SSOM: SS output management in master mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is used in master mode when SSOE is enabled. It allows to configure SS output between two consecutive data transfers.</td>
</tr>
<tr>
<td></td>
<td>0: SS is kept at active level till data transfer is completed, it becomes inactive with EOT flag</td>
</tr>
<tr>
<td></td>
<td>1: SPI data frames are interleaved with SS non active pulses when MIDI[3:0]&gt;1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>SSOE: SS output enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is taken into account at master mode only</td>
</tr>
<tr>
<td></td>
<td>0: SS output is disabled and the SPI can work in multi-master configuration</td>
</tr>
<tr>
<td></td>
<td>1: SS output is enabled. The SPI cannot work in a multi-master environment. It forces the SS pin at inactive level after the transfer in according with SSOM, MIDI, MSSI, SSIOP bits setting</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>SSIOP: SS input/output polarity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: low level is active for SS signal</td>
</tr>
<tr>
<td></td>
<td>1: high level is active for SS signal</td>
</tr>
</tbody>
</table>

| Bit 27 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bit 26</th>
<th>SSM: software management of SS signal input</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: SS input value is determined by the SS PAD</td>
</tr>
<tr>
<td></td>
<td>1: SS input value is determined by the SSI bit</td>
</tr>
<tr>
<td></td>
<td>SS signal input has to be managed by software (SSM=1,SSI=1) when SS output mode is enabled (SSOE=1) at master mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 25</th>
<th>CPOL: clock polarity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: SCK signal is at 0 when idle</td>
</tr>
<tr>
<td></td>
<td>1: SCK signal is at 1 when idle</td>
</tr>
</tbody>
</table>
Bit 24 **CHFA**: clock phase  
0: the first clock transition is the first data capture edge  
1: the second clock transition is the first data capture edge  

Bit 23 **LSBFIRST**: data frame format  
0: MSB transmitted first  
1: LSB transmitted first  
*Note: This bit can be also used in PCM and I2S modes.*  

Bit 22 **MASTER**: SPI master  
0: SPI Slave  
1: SPI Master  

Bits 21:19 **SP[2:0]**: Serial protocol  
000: SPI Motorola  
001: SPI TI  
others: Reserved, must not be used  

Bits 18:17 **COMM[1:0]**: SPI communication mode  
00: full-duplex  
01: simplex transmitter  
10: simplex receiver  
11: half-duplex  

Bit 16 Reserved, must be kept at reset value.  

Bit 15 **IOSWP**: swap functionality of MISO and MOSI pins  
0: no swap  
1: MOSI and MISO are swapped  
When this bit is set, the function of MISO and MOSI pins alternate functions are inverted.  
Original MISO pin becomes MOSI and original MOSI pin becomes MISO.  
*Note: This bit can be also used in PCM and I2S modes.*  

Bits 14:8 Reserved, must be kept at reset value.  

Bits 7:4 **MIDI[3:0]**: master Inter-Data Idleness  
Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in master mode.  
0000: no delay  
0001: 1 clock cycle period delay  
...  
1111: 15 clock cycle periods delay  
*Note: This feature is not supported in TI mode.*  

Bits 3:0 **MSSI[3:0]**: master SS idleness  
Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of SS and first data transaction start in master mode when SSOE is enabled.  
0000: no extra delay  
0001: 1 clock cycle period delay added  
...  
1111: 15 clock cycle periods delay added  
*Note: This feature is not supported in TI mode.*
55.11.5 SPI/I2S interrupt enable register (SPI2S_IER)

Address offset: 0x10
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:11</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>TSERFIE: additional number of transactions reload interrupt enable</td>
<td>0: TSERF interrupt disabled, 1: TSERF interrupt enabled</td>
</tr>
<tr>
<td>9</td>
<td>MODFIE: mode fault interrupt enable</td>
<td>0: MODF interrupt disabled, 1: MODF interrupt enabled</td>
</tr>
<tr>
<td>8</td>
<td>TIFREIE: TIFRE interrupt enable</td>
<td>0: TIFRE interrupt disabled, 1: TIFRE interrupt enabled</td>
</tr>
<tr>
<td>7</td>
<td>CRCEIE: CRC error interrupt enable</td>
<td>0: CRC interrupt disabled, 1: CRC interrupt enabled</td>
</tr>
<tr>
<td>6</td>
<td>OVRIE: OVR interrupt enable</td>
<td>0: OVR interrupt disabled, 1: OVR interrupt enabled</td>
</tr>
<tr>
<td>5</td>
<td>UDRIE: UDR interrupt enable</td>
<td>0: UDR interrupt disabled, 1: UDR interrupt enabled</td>
</tr>
<tr>
<td>4</td>
<td>TXTFIE: TXTFIE interrupt enable</td>
<td>0: TXTF interrupt disabled, 1: TXTF interrupt enabled</td>
</tr>
<tr>
<td>3</td>
<td>EOTIE: EOT, SUSP and TXC interrupt enable</td>
<td>0: EOT/SUSP/TXC interrupt disabled, 1: EOT/SUSP/TXC interrupt enabled</td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.
Bit 2 **DXPIE**: DXP interrupt enabled
- DXPIE is set by software and cleared by TXTF flag set event.
  - 0: DXP interrupt disabled
  - 1: DXP interrupt enabled

Bit 1 **TXPIE**: TXP interrupt enable
- TXPIE is set by software and cleared by TXTF flag set event.
  - 0: TXP interrupt disabled
  - 1: TXP interrupt enabled

Bit 0 **RXPIE**: RXP Interrupt Enable
- 0: RXP interrupt disabled
- 1: RXP interrupt enabled

### 55.11.6 SPI/I2S status register (SPI2S_SR)

**Address offset**: 0x14

**Reset value**: 0x0000 1002

<table>
<thead>
<tr>
<th>Bit 31-16</th>
<th>CTSIZE[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 15:16</td>
<td>number of data frames remaining in current TSIZE session</td>
</tr>
<tr>
<td></td>
<td>The value is not quite reliable when traffic is ongoing on bus</td>
</tr>
</tbody>
</table>

**Bit 15** **RXWNE**: RxFIFO word not empty
- RxFIFO contains more than one 32-bit data
  - 0: less than 32-bit data frame available in RxFIFO
  - 1: at least one 32-bit data is available in the RxFIFO

**Bits 14:13** **RXPLVL[1:0]**: RxFIFO packing level
- Number of packed frames available in the last 32-bit word of the RxFIFO. This number of frames is read from the RxFIFO when RXWNE=0.
  - When frame size is smaller than or equal to 8-bit
    - 00: the number of frames in RxFIFO is 0 or a multiple of 4.
    - 01: 1 frame available when RXWNE=0
    - 10: 2 frames available when RXWNE=0
    - 11: 3 frames available when RXWNE=0
  - When the frame size is smaller than or equal to 16-bit, but larger than 8-bit, RXPLV[1:0] can take the values 0 or 1.
    - 00: the number of frames in RxFIFO is 0 or a multiple of 2.
    - 01: the number of frame in RxFIFO is odd. One 16-bit data is available when RXWNE=0
    - Other values forbidden.
  - When the frame size is greater than 16-bit, these bits read as 00.
    - 00: -
    - Other values forbidden.
Bit 12 **T XC**: TxFIFO transmission complete
This flag is changed by hardware.
When TSIZE=0 the TXC raises each time the TxFIFO becomes empty and there is no activity on the bus.
If TSIZE ->0 the TXC raises at the end of transfer no matter on TxFIFO occupancy.
When this flag is set, master transmission is completed. The flag is not reliable to consider end of master reception. When CRC mode is enabled, TXC is set only after the CRC transmission. TXC generates an interrupt when EOTIE is set.
0: Current data transaction is still ongoing, data is available in TxFIFO or last frame transmission is on going (including CRC).
1: Last TxFIFO or CRC frame transmission completed

Bit 11 **SUSP**: suspend
In Master mode, SUSP is set by hardware when a CSUSP request is done, as soon as the current frame is completed or at master automatic suspend receive mode (MASRX bit is set at SPI2S_CR1 register) on RxFIFO full condition.
SUSP generates an interrupt when EOTIE is set.
This bit is cleared by write 1 to SUSPC bit at SPI2S_IFCR
0: SPI not suspended (master mode active or other mode).
1: Master mode SUSPended (Current Frame completed)

Bit 10 **TSERF**: additional number of SPI data to be transacted was reload
This bit is cleared by write 1 to TSERFC bit at SPI2S_IFCR or by writing the TSER[15:0] (SPI_CR2) register
0: no acceptation
1: additional number of data accepted, current transaction continues

Bit 9 **MODF**: mode fault
0: no mode fault
1: mode fault detected
This bit is cleared by write 1 to MODFC bit at SPI2S_IFCR

Bit 8 **TIFRE**: TI frame format error
0: no TI Frame Error
1: TI Frame Error detected
This bit is cleared by write 1 to TIFREC bit at SPI2S_IFCR

Bit 7 **CRCE**: CRC error
0: no CRC error
1: CRC error detected
This bit is cleared by write 1 to CRCEC bit at SPI2S_IFCR

Bit 6 **OVR**: overrun
0: no overrun
1: overrun detected
This bit is cleared by write 1 to OVRC bit at SPI2S_IFCR

Bit 5 **UDR**: underrun at slave transmission mode
0: no underrun
1: underrun detected
This bit is cleared by write 1 to UDRC bit at SPI2S_IFCR

*Note: UDR flag applies to Slave mode only*
**55.11.7 SPI/I2S interrupt/status flags clear register (SPI2S_IFCR)**

Address offset: 0x18

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SUSP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TSERF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MODF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TIFRE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CRCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OVR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXTF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EOT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0: upload of TxFIFO is on-going or not started
1: TxFIFO upload is finished

TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.

This bit is cleared by software write 1 to TXTFC bit at SPI2S_IFCR

TXTF flag triggers an interrupt if TXTFIE bit is set.

TXTF setting clears the TXIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.

**Bit 3 EOT: end of transfer**

EOT is set by hardware as soon as a full transfer is complete, that is when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared by software write 1 to EOTC bit at SPI2S_IFCR.

EOT flag triggers an interrupt if EOTIE bit is set.

If DXP flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into RxFIFO in one-shot.

0: transfer is on-going or not started
1: transfer complete

In master, EOT event terminates the data transaction and handles SS output optionally. In both master and slave, the event handles CRC transaction.

**Bit 2 DXP: duplex packet**

0: Tx_FIFO is Full and/or Rx_FIFO is Empty
1: Both Tx_FIFO has space for write and Rx_FIFO contains for read a single packet at least

**DXP flag is set when both TXP and RXP flags are set.**

**Bit 1 TXP: Tx-packet space available**

0: there is not enough space to locate next data packet at Tx_FIFO
1: Tx_FIFO has enough free location to host 1 data packet

TXP flag is changed by hardware. It monitors overall space currently available at Tx_FIFO if SPI is enabled. It has to be checked once a complete data packet is stored at Tx_FIFO.

**Bit 0 RXP: Rx-packet available**

0: Rx_FIFO is empty or a not complete data packet is received
1: Rx_FIFO contains at least 1 data packet

RXP flag is changed by hardware. It monitors number of overall data currently available at Rx_FIFO if SPI is enabled. It has to be checked once a data packet is completely read out from Rx_FIFO.

---

Bit 4 **TXTF:** transmission transfer filled

0: upload of Tx_FIFO is on-going or not started
1: Tx_FIFO upload is finished

TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the Tx_FIFO.

This bit is cleared by software write 1 to TXTFC bit at SPI2S_IFCR

TXTF flag triggers an interrupt if TXTFIE bit is set.

TXTF setting clears the TXIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.
55.11.8 SPI/I2S transmit data register (SPI2S_TXDR)

Address offset: 0x20

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-24</th>
<th>Bit 23-16</th>
<th>Bit 15-8</th>
<th>Bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXDR[31:16]</td>
<td>w w w w w w w w w w w w w w</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15-8</th>
<th>Bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXDR[15:0]</td>
<td>w w w w w w w w w w w w w w</td>
</tr>
</tbody>
</table>

Bits 31:0 TXDR[31:0]: transmit data register

The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.

Note: data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.
halfword-wise (16 bit access) in this case 2 data-bytes or 1 halfword-data can be written by single access.
word-wise (32 bit access). In this case 4 data-bytes or 2 halfword-data or word-data can be written by single access.
55.11.9  **SPI/I2S receive data register (SPI2S_RXDR)**

Address offset: 0x30  
Reset value: 0x0000 0000

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

**RXDR[31:16]**

Bits 31:0 **RXDR[31:0]:** receive data register  
The register serves as an interface with RxFIFO. When it is read, RxFIFO is accessed.  
**Note:** Data is always right-aligned. Unused bits are read as zero when the register is read.  
**Writing to the register is ignored.**

**Note:** DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read  
by single access  
halfword-wise (16 bit access) in this case 2 data-bytes or 1 halfword-data can be read  
by single access  
word-wise (32 bit access). In this case 4 data-bytes or 2 halfword-data or word-data  
can be read by single access

55.11.10  **SPI polynomial register (SPI_CRCPOLY)**

Address offset: 0x40  
Reset value: 0x0000 0107

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

**CRCPOLY[31:16]**

Bits 31:0 **CRCPOLY[31:0]:** CRC polynomial register  
This register contains the polynomial for the CRC calculation.  
The default 9-bit polynomial setting 0x107 corresponds to default 8-bit setting of DSIZE. It is  
compatible with setting 0x07 used at some other ST products with fixed length of the  
polynomial string where the most significant bit of the string is always kept hidden.  
Length of the polynomial is given by the most significant bit of the value stored at this  
register. It has to be set greater than DSIZE. CRC33_17 bit has to be set additionally with  
CRCPOLY register when DSIZE is configured to maximum 32-bit or 16-bit size and CRC is  
enabled (to keep polynomial length greater than data size).  
Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is  
no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are  
always read zero while any write to them is ignored.
55.11.11 SPI transmitter CRC register (SPI_TXCRC)

Address offset: 0x44
Reset value: 0x0000 0000

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

Bits 31:0 **TXCRC[31:0]**: CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of SPI_CR1 is written to 1 or when a data block is transacted completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register. The number of bits considered at calculation depends on SPI_CRCPOLY register and CRCSIZE bits settings at SPI_CFG1 register.

*Note:* A read to this register when the communication is ongoing could return an incorrect value.

Not used for the I²S mode.

Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

55.11.12 SPI receiver CRC register (SPI_RXCRC)

Address offset: 0x48
Reset value: 0x0000 0000

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

Note: A read to this register when the communication is ongoing could return an incorrect value.
55.11.13 SPI underrun data register (SPI_UDRDR)

Address offset: 0x4C

Reset value: 0x0000 0000

Bits 31:0 RXCRC[31:0]: CRC register for receiver

When CRC calculation is enabled, the RXCRC[31:0] bits contain the computed CRC value of the subsequently received bytes. CRC calculation is initialized when the CRCEN bit of SPI_CR1 is written to 1 or when a data block is transacted completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on SPI_CRCPOLY register and CRCSIZE bits settings at SPI_CFG1 register.

Note: A read to this register when the communication is ongoing could return an incorrect value.
Not used for the I2S mode.

Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit.
There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Bits 31:0 UDRDR[31:0]: data at slave underrun condition

The register is taken into account at slave mode and at underrun condition only. The number of bits considered depends on DSIZE bit settings at SPI_CFG1 register. Underrun condition handling depends on setting if UDRDET and UDRCFG bits at SPI_CFG1 register. Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

55.11.14 SPI/I2S configuration register (SPI_I2SCFGR)

Address offset: 0x50

Reset value: 0x0000 0000

Note: All documented bits in this register must be configured when the I2S is disabled (SPE = 0). These bits are not used in SPI mode except for I2SMOD which needs to be set to 0 in SPI mode.
Bits 31:26  Reserved, must be kept at reset value.

Bit 25  **MCKOE**: master clock output enable
0: Master clock output is disabled
1: Master clock output is enabled

Bit 24  **ODD**: odd factor for the prescaler
0: Real divider value is = I2SDIV * 2
1: Real divider value is = (I2SDIV * 2) + 1
Refer to Section 55.9.9: Clock generator for details

Bits 23:16  **I2SDIV[7:0]**: I2S linear prescaler
I2SDIV can take any values except the value 1, when ODD is also equal to 1.
Refer to Section 55.9.9: Clock generator for details

Bit 15  Reserved, must be kept at reset value.

Bit 14  **DATFMT**: data format
0: the data inside the SPI2S_RXDR or SPI2S_TXDR are right aligned
1: the data inside the SPI2S_RXDR or SPI2S_TXDR are left aligned.

Bit 13  **WSINV**: Word select inversion
This bit is used to invert the default polarity of WS signal.
0: in I2S Philips standard, Left channel is transfered when WS is low, and right channel when WS is high.
In MSB or LSB justified mode, Left channel is transfered when WS is high, and right channel when WS is low.
In PCM mode the start of frame is indicated by a rising edge.
1: in I2S Philips standard, Left channel is transfered when WS is high, and right channel when WS is low.
In MSB or LSB justified mode, Left channel is transfered when WS is low, and right channel when WS is high.
In PCM mode the start of frame is indicated by a falling edge.

Bit 12  **FIXCH**: fixed channel length in slave
0: the channel length in slave mode is different from 16 or 32 bits (CHLEN not taken into account)
1: the channel length in slave mode is supposed to be 16 or 32 bits (according to CHLEN)

Bit 11  **CKPOL**: serial audio clock polarity
0: the signals generated by the SPI/I2S (i.e. SDO and WS) are changed on the falling edge of CK and the signals received by the SPI/I2S (i.e. SDI and WS) are read on the rising edge of CK.
1: the signals generated by the SPI/I2S (i.e. SDO and WS) are changed on the rising edge of CK and the signals received by the SPI/I2S (i.e. SDI and WS) are read on the falling edge of CK.

Bit 10  **CHLEN**: channel length (number of bits per audio channel)
0: 16-bit wide
1: 32-bit wide
The bit write operation has a meaning only if DATLEN = 00 otherwise the channel length is fixed to 32-bit by hardware whatever the value filled in.

Bits 9:8  **DATLEN[1:0]**: data length to be transferred
00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: not allowed
Bit 7  **PCMSYNC**: PCM frame synchronization  
0: short frame synchronization  
1: long frame synchronization  

Bit 6  Reserved, must be kept at reset value.  

Bits 5:4  **I2SSTD[1:0]**: I2S standard selection  
00: I2S Philips standard.  
01: MSB justified standard (left justified)  
10: LSB justified standard (right justified)  
11: PCM standard  
For more details on I2S standards, refer to *Section 55.9.5: Supported audio protocols*  

Bits 3:1  **I2SCFG[2:0]**: I2S configuration mode  
000: slave - transmit  
001: slave - receive  
010: master - transmit  
011: master - receive  
100: slave - full duplex  
101: master - full duplex  
others, not used  

Bit 0  **I2SMOD**: I2S mode selection  
0: SPI mode is selected  
1: I2S/PCM mode is selected
### 55.12 SPI register map and reset values

#### Table 401. SPI register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>SPI2S_CR1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>SPI_CR2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>SPI_CFG1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>SPI_CFG2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>SPI2S_IER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>SPI2S_SR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>SPI2S_IFCR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>SPI2S_TXDR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>SPI2S_RXDR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x30</td>
<td>SPI2S_RXDR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x34</td>
<td>SPI2S_RXDR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x40</td>
<td>SPI_CRCPOLY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44</td>
<td>SPI_TXCRC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x48</td>
<td>SPI_RXCRC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4C</td>
<td>SPI_UDRDR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes:
- **(1)** The CRC polynomial is a preset value that gets updated if the frame is CRC-checked. This can be read, but not written.
- **(2)** Reserved or not used.
Table 401. SPI register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x50   | SPI_I2SCFGR   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Reset value

1. Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored. Refer to Table register boundary addresses.

Refer to Section 2.3 on page 128 for the register boundary addresses.
56 Serial audio interface (SAI)

56.1 Introduction

The SAI interface (Serial audio interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many stereo or mono audio applications may be targeted. I2S standards, LSB or MSB-justified, PCM/DSP, TDM, and AC’97 protocols may be addressed for example. SPDIF output is offered when the audio block is configured as a transmitter.

To bring this level of flexibility and reconfigurability, the SAI contains two independent audio subblocks. Each block has its own clock generator and I/O line controller.

The SAI works in master or slave configuration. The audio subblocks are either receiver or transmitter and work synchronously or not (with respect to the other one).

The SAI can be connected with other SAIIs to work synchronously.
56.2 SAI main features

- Two independent audio subblocks which can be transmitters or receivers with their respective FIFO.
- 8-word integrated FIFOs for each audio subblock.
- Synchronous or asynchronous mode between the audio subblocks.
- Possible synchronization between multiple SAI.
- Master or slave configuration independent for both audio subblocks.
- Clock generator for each audio block to target independent audio frequency sampling when both audio subblocks are configured in master mode.
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit.
- Audio protocol: I2S, LSB or MSB-justified, PCM/DSP, TDM, AC’97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required.
- Up to 16 slots available with configurable size.
- Number of bits by frame can be configurable.
- Frame synchronization active level configurable (offset, bit length, level).
- First active bit position in the slot is configurable.
- LSB first or MSB first for data transfer.
- Mute mode.
- Stereo/Mono audio frame capability.
- Communication clock strobing edge configurable (SCK).
- Error flags with associated interrupts if enabled respectively.
  - Overrun and underrun detection,
  - Anticipated frame synchronization signal detection in slave mode,
  - Late frame synchronization signal detection in slave mode,
  - Codec not ready for the AC’97 mode in reception.
- Interrupt sources when enabled:
  - Errors,
  - FIFO requests.
- 2-channel DMA interface.

56.3 SAI implementation

<table>
<thead>
<tr>
<th>SAI features</th>
<th>SAI1</th>
<th>SAI2</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2S, LSB or MSB-justified, PCM/DSP, TDM, AC’97</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FIFO size</td>
<td>8 words</td>
<td>8 words</td>
</tr>
<tr>
<td>SPDIF</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PDM</td>
<td>X(2)</td>
<td>-</td>
</tr>
</tbody>
</table>

1. ‘X’ = supported, ‘-’ = not supported.
2. Only signals D[3:1], and CK[2:1] are available.

Table 402. STM32H7A3/7B3 and STM32H7B0 SAI interfaces (1)
56.4 SAI functional description

56.4.1 SAI block diagram

"Figure 611" shows the SAI block diagram while "Table 403" and "Table 404" list SAI internal and external signals.

The SAI is mainly composed of two audio subblocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA in order to leave the CPU free during the communication. Each audio block is independent. They can be synchronous with each other.

An I/O line controller manages a set of 4 dedicated pins (SD, SCK, FS, MCLK) for a given audio block in the SAI. Some of these pins can be shared if the two subblocks are declared as synchronous to leave some free to be used as general purpose I/Os. The MCLK pin can be output, or not, depending on the application, the decoder requirement and whether the audio block is configured as the master.

If one SAI is configured to operate synchronously with another one, even more I/Os can be freed (except for pins SD_x).
The functional state machine can be configured to address a wide range of audio protocols. Some registers are present to set-up the desired protocols (audio frame waveform generator).

The audio subblock can be a transmitter or receiver, in master or slave mode. The master mode means the SCK\_x bit clock and the frame synchronization signal are generated from the SAI, whereas in slave mode, they come from another external or internal master. There is a particular case for which the FS signal direction is not directly linked to the master or slave mode definition. In AC’97 protocol, it will be an SAI output even if the SAI (link controller) is set-up to consume the SCK clock (and so to be in Slave mode).

Note: For ease of reading of this section, the notation SAI\_x refers to SAI\_A or SAI\_B, where ‘x’ represents the SAI A or B subblock.

56.4.2 SAI pins and internal signals

### Table 403. SAI internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sai_a_gbl_it/ saib_gbl_it</td>
<td>Output</td>
<td>Audio block A and B global interrupts.</td>
</tr>
<tr>
<td>sai_a_dma, saib_dma</td>
<td>Input/output</td>
<td>Audio block A and B DMA acknowledges and requests.</td>
</tr>
<tr>
<td>sai_sync_out_sck, sai_sync_out_fs</td>
<td>Output</td>
<td>Internal clock and frame synchronization output signals exchanged with other SAI blocks.</td>
</tr>
<tr>
<td>sai_sync_in_sck, sai_sync_in_fs</td>
<td>Input</td>
<td>Internal clock and frame synchronization input signals exchanged with other SAI blocks.</td>
</tr>
<tr>
<td>sai_a_ker_ck/ saib_ker_ck</td>
<td>Input</td>
<td>Audio block A/B kernel clock.</td>
</tr>
<tr>
<td>sai_pclk</td>
<td>Input</td>
<td>APB clock.</td>
</tr>
</tbody>
</table>

### Table 404. SAI input/output pins

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAI_SCK_A/B</td>
<td>Input/output</td>
<td>Audio block A/B bit clock.</td>
</tr>
<tr>
<td>SAI_MCLK_A/B</td>
<td>Output</td>
<td>Audio block A/B master clock.</td>
</tr>
<tr>
<td>SAI_SD_A/B</td>
<td>Input/output</td>
<td>Data line for block A/B.</td>
</tr>
<tr>
<td>SAI_FS_A/B</td>
<td>Input/output</td>
<td>Frame synchronization line for audio block A/B.</td>
</tr>
<tr>
<td>SAI_CK[4:1]</td>
<td>Output</td>
<td>PDM bitstream clock(^\d).</td>
</tr>
<tr>
<td>SAI_D[4:1]</td>
<td>Input</td>
<td>PDM bitstream data(^\d).</td>
</tr>
</tbody>
</table>

1. These signals might not be available in all SAI instances. Please refer to Section 56.3: SAI implementation for details.
56.4.3 Main SAI modes

Each audio subblock of the SAI can be configured to be master or slave via MODE bits in the SAI_xCR1 register of the selected audio block.

Master mode

In master mode, the SAI delivers the timing signals to the external connected device:

- The bit clock and the frame synchronization are output on pin SCK_x and FS_x, respectively.
- If needed, the SAI can also generate a master clock on MCLK_x pin.

Both SCK_x, FS_x and MCLK_x are configured as outputs.

Slave mode

The SAI expects to receive timing signals from an external device.

- If the SAI subblock is configured in asynchronous mode, then SCK_x and FS_x pins are configured as inputs.
- If the SAI subblock is configured to operate synchronously with another SAI interface or with the second audio subblock, the corresponding SCK_x and FS_x pins are left free to be used as general purpose I/Os.

In slave mode, MCLK_x pin is not used and can be assigned to another function.

It is recommended to enable the slave device before enabling the master.

Configuring and enabling SAI modes

Each audio subblock can be independently defined as a transmitter or receiver through the MODE bit in the SAI_xCR1 register of the corresponding audio block. As a result, SAI_SD_x pin will be respectively configured as an output or an input.

Two master audio blocks in the same SAI can be configured with two different MCLK and SCK clock frequencies. In this case they have to be configured in asynchronous mode.

Each of the audio blocks in the SAI are enabled by SAIEN bit in the SAI_xCR1 register. As soon as this bit is active, the transmitter or the receiver is sensitive to the activity on the clock line, data line and synchronization line in slave mode.

In master TX mode, enabling the audio block immediately generates the bit clock for the external slaves even if there is no data in the FIFO. However FS signal generation is conditioned by the presence of data in the FIFO. After the FIFO receives the first data to transmit, this data is output to external slaves. If there is no data to transmit in the FIFO, 0 values are then sent in the audio frame with an underrun flag generation.

In slave mode, the audio frame starts when the audio block is enabled and when a start of frame is detected.

In Slave TX mode, no underrun event is possible on the first frame after the audio block is enabled, because the mandatory operating sequence in this case is:

1. Write into the SAI_xDR (by software or by DMA).
2. Wait until the FIFO threshold (FLH) flag is different from 0b000 (FIFO empty).
3. Enable the audio block in slave transmitter mode.
56.4.4 **SAI synchronization mode**

There are two levels of synchronization, either at audio subblock level or at SAI level.

**Internal synchronization**

An audio subblock can be configured to operate synchronously with the second audio subblock in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK_x, FS_x, and MCLK_x pins released back as GPIOs while the audio block configured in asynchronous mode is the one for which FS_x and SCK_x ad MCLK_x I/O pins are relevant (if the audio block is considered as master).

Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding SYNCEN[1:0] bits set to 00 in SAI_xCR1) and one synchronous block (corresponding SYNCEN[1:0] bits set to 01 in the SAI_xCR1).

**Note:** Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.

**External synchronization**

The audio subblocks can also be configured to operate synchronously with another SAI. This can be done as follow:

1. The SAI, which is configured as the source from which the other SAI is synchronized, has to define which of its audio subblock is supposed to provide the FS and SCK signals to other SAI. This is done by programming SYNCOUT[1:0] bits.

2. The SAI which shall receive the synchronization signals has to select which SAI will provide the synchronization by setting the proper value on SYNCIN[1:0] bits. For each of the two SAI audio subblocks, the user must then specify if it operates synchronously with the other SAI via the SYNCEN bit.

**Note:** SYNCIN[1:0] and SYNCOUT[1:0] bits are located into the SAI_GCR register, and SYNCEN bits into SAI_xCR1 register.

If both audio subblocks in a given SAI need to be synchronized with another SAI, it is possible to choose one of the following configurations:

- Configure each audio block to be synchronous with another SAI block through the SYNCEN[1:0] bits.
- Configure one audio block to be synchronous with another SAI through the SYNCEN[1:0] bits. The other audio block is then configured as synchronous with the second SAI audio block through SYNCEN[1:0] bits.

The following table shows how to select the proper synchronization signal depending on the SAI block used. For example SAI2 can select the synchronization from SAI1 by setting SAI2 SYNCIN to 0. If SAI1 wants to select the synchronization coming from SAI2, SAI1 SYNCIN must be set to 1. Positions noted as ‘Reserved’ shall not be used.
56.4.5 Audio data size

The audio frame can target different data sizes by configuring bit DS[2:0] in the SAI_xCR1 register. The data sizes may be 8, 10, 16, 20, 24 or 32 bits. During the transfer, either the MSB or the LSB of the data are sent first, depending on the configuration of bit LSBFIRST in the SAI_xCR1 register.

56.4.6 Frame synchronization

The FS signal acts as the Frame synchronization signal in the audio frame (start of frame). The shape of this signal is completely configurable in order to target the different audio protocols with their own specificities concerning this Frame synchronization behavior. This reconfigurability is done using register SAI_xFRCR. Figure 612 illustrates this flexibility.

Table 405. External synchronization selection

<table>
<thead>
<tr>
<th>Block instance</th>
<th>SYNCIN= 1</th>
<th>SYNCIN= 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAI1</td>
<td>SAI2 sync.</td>
<td>Reserved</td>
</tr>
<tr>
<td>SAI2</td>
<td>Reserved</td>
<td>SAI1 sync.</td>
</tr>
</tbody>
</table>

In AC'97 mode or in SPDIF mode (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01 in the SAI_xCR1 register), the frame synchronization shape is forced to match the AC’97 protocol. The SAI_xFRCR register value is ignored.

Each audio block is independent and consequently each one requires a specific configuration.

Frame length

- Master mode

The audio frame length can be configured to up to 256 bit clock cycles, by setting FRL[7:0] field in the SAI_xFRCR register.

If the frame length is greater than the number of declared slots for the frame, the remaining bits to transmit will be extended to 0 or the SD line will be released to HI-z
depending the state of bit TRIS in the SAI_xCR2 register (refer to FS signal role). In reception mode, the remaining bit is ignored.
If bit NODIV is cleared, (FRL+1) must be equal to a power of 2, from 8 to 256, to ensure that an audio frame contains an integer number of MCLK pulses per bit clock cycle.
If bit NODIV is set, the (FRL+1) field can take any value from 8 to 256. Refer to Section 56.4.8: SAI clock generator.

- Slave mode
  The audio frame length is mainly used to specify to the slave the number of bit clock cycles per audio frame sent by the external master. It is used mainly to detect from the master any anticipated or late occurrence of the Frame synchronization signal during an on-going audio frame. In this case an error will be generated. For more details refer to Section 56.4.14: Error flags.
  In slave mode, there are no constraints on the FRL[7:0] configuration in the SAI_xFRCR register.

The number of bits in the frame is equal to FRL[7:0] + 1.
The minimum number of bits to transfer in an audio frame is 8.

**Frame synchronization polarity**

FSPOL bit in the SAI_xFRCR register sets the active polarity of the FS pin from which a frame is started. The start of frame is edge sensitive.
In slave mode, the audio block waits for a valid frame to start transmitting or receiving. Start of frame is synchronized to this signal. It is effective only if the start of frame is not detected during an ongoing communication and assimilated to an anticipated start of frame (refer to Section 56.4.14: Error flags).
In master mode, the frame synchronization is sent continuously each time an audio frame is complete until the SAIEN bit in the SAI_xCR1 register is cleared. If no data are present in the FIFO at the end of the previous audio frame, an underrun condition will be managed as described in Section 56.4.14: Error flags, but the audio communication flow will not be interrupted.

**Frame synchronization active level length**

The FSALL[6:0] bits of the SAI_xFRCR register allow configuring the length of the active level of the Frame synchronization signal. The length can be set from 1 to 128 bit clock cycles.
As an example, the active length can be half of the frame length in I2S, LSB or MSB-justified modes, or one-bit wide for PCM/DSP or TDM.

**Frame synchronization offset**

Depending on the audio protocol targeted in the application, the Frame synchronization signal can be asserted when transmitting the last bit or the first bit of the audio frame (this is the case in I2S standard protocol and in MSB-justified protocol, respectively). FSOFF bit in the SAI_xFRCR register allows to choose one of the two configurations.
FS signal role

The FS signal can have a different meaning depending on the FS function. FSDEF bit in the SAI_xFRCR register selects which meaning it will have:

- 0: start of frame, like for instance the PCM/DSP, TDM, AC’97, audio protocols,
- 1: start of frame and channel side identification within the audio frame like for the I2S, the MSB or LSB-justified protocols.

When the FS signal is considered as a start of frame and channel side identification within the frame, the number of declared slots must be considered to be half the number for the left channel and half the number for the right channel. If the number of bit clock cycles on half audio frame is greater than the number of slots dedicated to a channel side, and TRIS = 0, 0 is sent for transmission for the remaining bit clock cycles in the SAI_xCR2 register. Otherwise if TRIS = 1, the SD line is released to HI-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.

Figure 613. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)

1. The frame length should be even.

If FSDEF bit in SAI_xFRCR is kept clear, so FS signal is equivalent to a start of frame, and if the number of slots defined in NBSLOT[3:0] in SAI_xSLOTTR multiplied by the number of bits by slot configured in SLOTSZ[1:0] in SAI_xSLOTTR is less than the frame size (bit FRL[7:0] in the SAI_xFRCR register), then:
if TRIS = 0 in the SAI\_xCR2 register, the remaining bit after the last slot will be forced to 0 until the end of frame in case of transmitter,
if TRIS = 1, the line will be released to HI-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.

**Figure 614. FS role is start of frame (FSDEF = 0)**

The FS signal is not used when the audio block in transmitter mode is configured to get the SPDIF output on the SD line. The corresponding FS I/O will be released and left free for other purposes.

### 56.4.7 Slot configuration

The slot is the basic element in the audio frame. The number of slots in the audio frame is equal to NBSLOT[3:0] + 1.

The maximum number of slots per audio frame is fixed at 16.

For AC’97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI\_xSLOTR register.

When a invalid slot is transferred, the SD data line is either forced to 0 or released to HI-z depending on TRIS bit configuration (refer to Output data line management on an inactive slot) in transmitter mode. In receiver mode, the received value from the end of this slot is ignored. Consequently, there will be no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

The slot size is also configurable as shown in Figure 615. The size of the slots is selected by setting SLOTSZ[1:0] bits in the SAI\_xSLOTTR register. The size is applied identically for each slot in an audio frame.
If it is possible to choose the position of the first data bit to transfer within the slots. This offset is configured by FBOFF[4:0] bits in the SAI_xSLOTR register. 0 values will be injected in transmitter mode from the beginning of the slot until this offset position is reached. In reception, the bit in the offset phase is ignored. This feature targets the LSB justified protocol (if the offset is equal to the slot size minus the data size).

It is mandatory to respect the following conditions to avoid bad SAI behavior:

- FBOFF ≤ (SLOTSZ - DS),
- DS ≤ SLOTSZ,
- NBSLOT x SLOTSZ ≤ FRL (frame length).

The number of slots must be even when bit FSDEF in the SAI_xFRCR register is set.

In AC'97 and SPDIF protocol (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the slot size is automatically set as defined in Section 56.4.11: AC'97 link controller.
56.4.8 SAI clock generator

Each audio block has its own clock generator. The clock generator builds the master clock (MCLK\_x) and bit clock (SCK\_x) signals from the sai\_x\_ker\_ck. The sai\_x\_ker\_ck clock is delivered by the clock controller of the product (RCC).

**Generation of the master clock (MCLK\_x)**

The clock generator provides the master clock (MCLK\_x) when the audio block is defined as Master or Slave. The master clock is generated as soon as the MCKEN bit is set to 1 even if the SAIEN bit for the corresponding block is set to 0. This feature can be useful if the MCLK\_x clock is used as system clock for an external audio device, since it allows generating the MCLK\_x before activating the audio stream.

To generate a master clock on MCLK\_x output before transferring the audio samples, the user application has to follow the sequence below:

1. Check that SAIEN = 0.
2. Program the MCKDIV[5:0] divider to the required value.
3. Set the MCKEN bit to 1.
4. Later, the application can configure other parts of the SAI, and sets the SAIEN bit to 1 to start the transfer of audio samples.

To avoid disturbances on the clock generated on MCLK\_x output, the following operations are not recommended:

- Changing MCKDIV when MCKEN = 1
- Setting MCKEN to 0 if the SAIEN = 1

The SAI guarantees that there is no spurs on MCLK\_x output when the MCLK\_x is switched ON and OFF via MCKEN bit (with SAIEN = 0).

*Table 406* shows MCLK\_x activation conditions.

<table>
<thead>
<tr>
<th>MCLKEN</th>
<th>NODIV</th>
<th>SAIEN for block x</th>
<th>MCLK_x</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Enabled</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Disabled</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Enabled</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td></td>
<td>Enabled</td>
</tr>
</tbody>
</table>

*Note:* MCLK\_x can also be generated in AC'97 mode, when MCLKEN is set to 1.
Generation of the bit clock (SCK_x)

The clock generator provides the bit clock (SCK_x) when the audio block is defined as Master. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator.

In Slave mode, the value of NODIV and OSR fields are ignored, and the SCK_x clock is not generated.

The bit clock strobing edge of SCK_x can be configured through the CKSTR fields, which is functional both in master and slave mode.

*Figure 617* illustrates the architecture of the audio block clock generator.

![Figure 617. Audio block clock generator overview](image)

The NODIV bit must be used to force the ratio between the master clock (MCLK_x) and the frame synchronization (FS_x) frequency to 256 or 512.

- If NODIV is set to 0, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, according to OSR value, but the frame length must be a power of 2. More details are given hereafter.
- If NODIV is set to 1, the application can adjust the frequency of the bit clock (SCK_x) via MCKDIV. In addition there is no restriction on the frame length value as long as the frame length is bigger or equal to 8 (i.e. FRL[7:0] > 6). The frame synchronization frequency depends on MCKDIV and frame length (FRL[7:0]). In that case, the frequency of the MCLK_x is equal to the SCK_x.

The NODIV, MCKEN, SAIEN, OVR, CKSTR and MCKDIV[5:0] bits belong to the SAI_xCR1 register, while FRL[7:0] belongs to SAI_xFRCR.
Clock generator programming when NODIV = 0

In that case, MCLK_x frequency will be:

- \( F_{MCLK_x} = 256 \times F_{FS_x} \) if OSR = 0
- \( F_{MCLK_x} = 512 \times F_{FS_x} \) if OSR = 1

When MCKDIV is different from 0, MCLK_x frequency is given by the formula below:

\[
F_{MCLK_x} = \frac{F_{sai_x\_ker\_ck}}{MCKDIV}
\]

The frame synchronization frequency is given by:

\[
F_{FS_x} = \frac{F_{sai_x\_ker\_ck}}{MCKDIV \times (OSR + 1) \times 256}
\]

The bit clock frequency (SCK_x) is given by the following formula:

\[
F_{SCK_x} = \frac{F_{sai_x\_ker\_ck} \times (FRL + 1)}{MCKDIV \times (OSR + 1) \times 256}
\]

**Note:**
When NODIV is equal to 0, \((FRL+1)\) must be a power of two. In addition \((FRL+1)\) must range between 8 and 256. \((FRL +1)\) represents the number of bit clock in the audio frame.

When MCKDIV division ratio is odd, the MCLK duty cycle will not be 50%. The bit clock signal (SCK_x) can also have a duty cycle different from 50% if MCKDIV is odd, if OSR is equal to 0, and if \((FRL+1) = 2^8\).

It is recommended, to program MCKDIV to an even value or to big values (higher than 10).

**Note that MCKDIV = 0 gives the same result as MCKDIV = 1.**

Clock generator programming when NODIV = 1

When MCKDIV is different from 0, the frequency of the bit clock (SCK_x) is given in the formula below:

\[
F_{SCK_x} = F_{MCLK_x} = \frac{F_{sai_x\_ker\_ck}}{MCKDIV}
\]

The frequency of the frame synchronization (FS_x) is given by the following formula:

\[
F_{FS_x} = \frac{F_{sai_x\_ker\_ck}}{(FRL + 1) \times MCKDIV}
\]

**Note:**
When NODIV is set to 1, \((FRL+1)\) can take any values from 8 to 256.

**Note that MCKDIV = 0 gives the same result as MCKDIV = 1.**
56.4.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. There is therefore only one FIFO request linked to FREQ bit in the SAI_xSR register.

An interrupt is generated if FREQIE bit is enabled in the SAI_xIM register. This depends on:
- FIFO threshold setting (FLVL bits in SAI_xCR2)
- Communication direction (transmitter or receiver). Refer to Interrupt generation in transmitter mode and Interrupt generation in reception mode.

Interrupt generation in transmitter mode

The interrupt generation depends on the FIFO configuration in transmitter mode:
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit set by hardware to 1 in SAI_xSR register) if no data are available in SAI_xDR register (FLVL[2:0] bits in SAI_xSR is less than 001b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO is no more empty (FLVL[2:0] bits in SAI_xSR are different from 0b000) i.e one or more data are stored in the FIFO.
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 001b), an interrupt is generated (FREQ bit set by hardware to 1 in SAI_xSR register) if less than a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR is less than 0b010). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when at least a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher or equal to 0b010).
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit set by hardware to 1 in

### Clock generator programming examples

Table 407 gives programming examples for 48, 96 and 192 kHz.

#### Table 407. Clock generator programming examples

<table>
<thead>
<tr>
<th>Input saix_ker_ck clock frequency</th>
<th>MCLK</th>
</tr>
</thead>
<tbody>
<tr>
<td>98.304 MHz</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input saix_ker_ck clock frequency</th>
<th>MCLK</th>
<th>F_MCLK/FS</th>
<th>FRL (1)</th>
<th>OSR</th>
<th>NODIV</th>
<th>MCKEN</th>
<th>MCKDIV[5:0]</th>
<th>Audio Sampling frequency (FS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>2^N-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0 or 1</td>
<td>1</td>
<td>1</td>
<td>192 kHz</td>
</tr>
<tr>
<td>512</td>
<td>2^N-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>96 kHz</td>
</tr>
<tr>
<td>512</td>
<td>2^N-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>48 kHz</td>
</tr>
<tr>
<td>256</td>
<td>2^N-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>192 kHz</td>
</tr>
<tr>
<td>256</td>
<td>2^N-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>96 kHz</td>
</tr>
<tr>
<td>256</td>
<td>2^N-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>48 kHz</td>
</tr>
<tr>
<td>63</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>192 kHz</td>
</tr>
<tr>
<td>63</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>48 kHz</td>
</tr>
</tbody>
</table>

1. N is an integer value between 3 and 8.
SAI_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 011b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when at least half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher or equal to 011b).

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO three quarter full (FTH[2:0] set to 011b), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if less than three quarters of the FIFO contain data (FLVL[2:0] bits in SAI_xSR are less than 0b100). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when at least three quarters of the FIFO contain data (FLVL[2:0] bits in SAI_xSR are higher or equal to 0b100).

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI_xSR is less than 101b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR is equal to 101b value).

### Interrupt generation in reception mode

The interrupt generation depends on the FIFO configuration in reception mode:

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least one data is available in SAI_xDR register (FLVL[2:0] bits in SAI_xSR is higher or equal to 001b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in SAI_xSR is equal to 0b000) i.e no data are stored in FIFO.

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO quarter fully full (FTH[2:0] set to 001b), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR is higher or equal to 0b010). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when less than a quarter of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR is less than 0b010).

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO half fully (FTH[2:0] set to 0b010 value), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR is higher or equal to 011b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when less than half of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR is less than 011b).

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO three quarter full (FTH[2:0] set to 011b value), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR is higher or equal to 0b100). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations available (FLVL[2:0] bits in SAI_xSR is less than 0b100).

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI_xSR is equal to 101b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO is not full (FLVL[2:0] bits in SAI_xSR is less than 101b).
Like interrupt generation, the SAI can use the DMA if DMAEN bit in the SAI_xCR1 register is set. The FREQ bit assertion mechanism is the same as the interrupt generation mechanism described above for FREQIE.

Each FIFO is an 8-word FIFO. Each read or write operation from/to the FIFO targets one word FIFO location whatever the access size. Each FIFO word contains one audio slot. FIFO pointers are incremented by one word after each access to the SAI_xDR register.

Data should be right aligned when it is written in the SAI_xDR.

Data received will be right aligned in the SAI_xDR.

The FIFO pointers can be reinitialized when the SAI is disabled by setting bit FFLUSH in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled the data present in the FIFO will be lost automatically.

56.4.10 PDM Interface

The PDM (Pulse Density Modulation) interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. Figure 618 shows a typical connection of a digital microphone pair via a PDM interface. Both microphones share the same bitstream clock and data line. Thanks to a configuration pin (LR), a microphone can provide valid data on SAI_CK[m] rising edge while the other provides valid data on SAI_CK[m] falling edge (m being the number of clock lines).

Figure 618. PDM typical connection and timing

1. \( n \) refers to the number of data lines and \( p \) to the number of microphone pairs.

The PDM function is intended to be used in conjunction with SAI_A subblock configured in TDM master mode. It cannot be used with SAI_B subblock. The PDM interface uses the timing signals provided by the TDM interface of SAI_A and adapts them to generate a bitstream clock (SAI_CK[m]).
The data processing sequence into the PDM is the following:
1. The PDM interface builds the bitstream clock from the bit clock received from the TDM interface of SAI_A.
2. The bitstream data received from the microphones (SAI_D[n]) are de-interleaved and go through a 7-bit delay line in order to fine-tune the delay of each microphone with the accuracy of the bitstream clock.
3. The shift registers translate each serial bitstream into bytes.
4. The last operation consists in shifting-out the resulting bytes to SAI_A via the serial data line of the TDM interface.

Figure 619 hereafter shows the block diagram of PDM interface, with a detailed view of a de-interleaver.

Note: The PDM interface does not embed the decimation filter required to build-up the PCM audio samples from the bitstream. It is up to the application software to perform this operation.

Figure 619. Detailed PDM interface block diagram

1. n refers to the number of data lines and p to the number of microphone pairs.

The PDM interface can be enabled through the PDMEN bit in SAI_PDMCR register. However the PDM interface must be enabled prior to enabling SAI_A block.

To reduce the memory footprint, the user can select the amount of microphones the application needs. This can be done through MICNBR[1:0] bits. It is possible to select between 2, 4, 6 or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.
Enabling the PDM interface

To enable the PDM interface, follow the sequence below:
1. Configure SAI_A in TDM master mode (see Table 408).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via PDMEN bit.
4. Enable the SAI_A.

Note: Once the PDM interface and SAI_A are enabled, the first 2 TDMA frames received on SAI_ADR are invalid and shall be dropped.

Start-up sequence

Figure 620 shows the start-up sequence: Once the PDM interface is enabled, it waits for the frame synchronization event prior to starting the acquisition of the microphone samples. After 8 SAI_CK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the TDM interface.

SAI_ADR data format

The arrangement of the data coming from the microphone into the SAI_ADR register depends on the following parameters:
- The amount of microphones
- The slot width selected
- LSBFIRST bit.

The slot width defines the amount of significant bits into each word available into the SAI_ADR.

When a slot width of 32 bits is selected, each data available into the SAI_ADR will contain 32 useful bits. This reduces the amount of words stored into the memory. However the...
counterpart is that the software has to perform some operations to de-interleave the data of each microphone.

In the other hand, when the slot width is set to 8 bits, each data available into the SAI_ADR will contain 8 useful bits. This increases the amount of words stored into the memory. However, it offers the advantage to avoid extra processing since each word contains information from one microphone.

**SAI_ADR data format example**

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to Figure 621.
  
  For an 8 microphone configuration, two consecutive words read from the SAI_ADR register contain a data byte from each microphone.

  For a 4 microphones configuration, each word read from the SAI_ADR register contains a data byte from each microphone.

  ![Figure 621. SAI_ADR format in TDM, 32-bit slot width](image)

- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to Figure 622.

  For an 8 microphone configuration, four consecutive words read from the SAI_ADR register contain a data byte from each microphone. Note that the 16-bit data of SAI_ADR are right aligned.

  For 4 or 2 microphone configuration, the SAI behavior is similar to 8-microphone configurations. Up to 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.
### Figure 622. SAI_ADR format in TDM, 16-bit slot width

<table>
<thead>
<tr>
<th>8 Microphones configuration</th>
<th>4 Microphones configuration</th>
<th>2 Microphones configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>word 4n</td>
<td>word 2n</td>
<td>word 2n</td>
</tr>
<tr>
<td>zeros</td>
<td>zeros</td>
<td>zeros</td>
</tr>
<tr>
<td>word 4n+1</td>
<td>word 2n+1</td>
<td>word 2n+1</td>
</tr>
<tr>
<td>zeros</td>
<td>zeros</td>
<td></td>
</tr>
<tr>
<td>word 4n+3</td>
<td></td>
<td>zeros</td>
</tr>
<tr>
<td></td>
<td></td>
<td>word 2n+2</td>
</tr>
</tbody>
</table>

#### Using a 8-bit slot width (DS = 0b010 and SLOTSZ = 0).

For an 8 microphone configuration, 8 consecutive words read from the SAI_ADR register contain a byte of data from each microphone. Note that the 8-bit data of SAI_ADR are right aligned.

For 4 or 2 microphone configuration, the SAI behavior is similar to 8 microphone configurations. Up to 4 words of 8 bits are required to acquire a byte from 4 microphones and 2 words from 2 microphones.
TDM configuration for PDM interface

SAI_A TDM interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the PDM interface as shown in Table 408 to ensure a good connection with the PDM interface.

Table 408. TDM settings

<table>
<thead>
<tr>
<th>Bit Fields</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE</td>
<td>0b01</td>
<td>Mode must be MASTER receiver</td>
</tr>
<tr>
<td>PRTCFG</td>
<td>0b00</td>
<td>Free protocol for TDM</td>
</tr>
<tr>
<td>DS</td>
<td>X</td>
<td>To be adjusted according to the required data format, in accordance to the frame length and the number of slots (FRL and NBSLOT). See Table 409.</td>
</tr>
<tr>
<td>LSBFIRST</td>
<td>X</td>
<td>This parameter can be used according to the wanted data format</td>
</tr>
<tr>
<td>CKSTR</td>
<td>0</td>
<td>Signal transitions occur on the rising edge of the SCK_A bit clock. Signals are stable on the falling edge of the bit clock.</td>
</tr>
<tr>
<td>MONO</td>
<td>0</td>
<td>Stereo mode</td>
</tr>
<tr>
<td>FRL</td>
<td>X</td>
<td>To be adjusted according to the number of microphones (MICNBR). See Table 409.</td>
</tr>
<tr>
<td>FSALL</td>
<td>0</td>
<td>Pulse width is one bit clock cycle</td>
</tr>
<tr>
<td>FSDEF</td>
<td>0</td>
<td>FS signal is a start of frame</td>
</tr>
</tbody>
</table>
Adjusting the bitstream clock rate

To properly program the SAI TDM interface, the user application must take into account the settings given in Table 408, and follow the below rules:

1. Adjust the bit clock frequency ($F_{SCK_A}$) according to the required frequency for the PDM bitstream clock, using the following formula:

   $$ F_{SCK_A} = F_{PDM_CK} \times (MICNBR + 1) \times 2 $$

   MICNBR can be 0, 1, or 2 ($0 = 2$ microphones, see Section 56.6.18)

2. Set the frame length (FRL) using the following formula

   $$ FRL = (16 \times (MICNBR + 1)) - 1 $$

3. Configure the slot size (DS) to a multiple of (FRL+1).

<table>
<thead>
<tr>
<th>Bit Fields</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSPOH</td>
<td>1</td>
<td>FS is active High</td>
</tr>
<tr>
<td>FSOFF</td>
<td>0</td>
<td>FS is asserted on the first bit of slot 0</td>
</tr>
<tr>
<td>FBOFF</td>
<td>0</td>
<td>No offset on slot</td>
</tr>
<tr>
<td>SLOTSZ</td>
<td>0</td>
<td>Slot size = data size</td>
</tr>
<tr>
<td>NBSLOT</td>
<td>X</td>
<td>To be adjusted according to the required data format, in accordance to the slot size, and the frame length (FRL and DS). See Table 409.</td>
</tr>
<tr>
<td>SLOTEN</td>
<td>X</td>
<td>To be adjusted according to NBSLOT</td>
</tr>
<tr>
<td>NODIV</td>
<td>1</td>
<td>No need to generate a master clock MCLK</td>
</tr>
<tr>
<td>MCKDIV</td>
<td>X</td>
<td>Depends on the frequency provided to sai_a_ker_ck input. This parameter shall be adjusted to generate the proper bitstream clock frequency. See Table 409.</td>
</tr>
</tbody>
</table>
Adjusting the delay lines

When the PDM interface is enabled, the application can adjust on-the-fly the delay cells of each microphone input via SAI_PDMDLY register.

The new delays values will become effective after two TDM frames.
56.4.11 AC’97 link controller

The SAI is able to work as an AC’97 link controller. In this protocol:

- The slot number and the slot size are fixed.
- The frame synchronization signal is perfectly defined and has a fixed shape.

To select this protocol, set PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed to 13 slots. The first one is 16-bit wide and all the others are 20-bit wide (data slots).
- FBOFF[4:0] bits in the SAI_xSLOTR register are ignored.
- The SAI_xFRCR register is ignored.
- The MCLK is not used.

The FS signal from the block defined as asynchronous is configured automatically as an output, since the AC’97 controller link drives the FS signal whatever the master or slave configuration.

*Figure 624* shows an AC’97 audio frame structure.

**Figure 624. AC’97 audio frame**

Note: In AC’97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC’97 protocol standard.

One SAI can be used to target an AC’97 point-to-point communication.

Using two SAIs (for devices featuring two embedded SAIs) allows controlling three external AC’97 decoders as illustrated in *Figure 625.*

In SAI1, the audio block A must be declared as asynchronous master transmitter whereas the audio block B is defined to be slave receiver and internally synchronous to the audio block A.

The SAI2 is configured for audio block A and B both synchronous with the external SAI1 in slave receiver mode.
In receiver mode, the SAI acting as an AC’97 link controller requires no FIFO request and so no data storage in the FIFO when the Codec ready bit in the slot 0 is decoded low. If bit CNRDYIE is enabled in the SAI_xIM register, flag CNRDY will be set in the SAI_xSR register and an interrupt is generated. This flag is dedicated to the AC’97 protocol.

**Clock generator programming in AC’97 mode**

In AC’97 mode, the frame length is fixed at 256 bits, and its frequency shall be set to 48 kHz. The formulas given in Section 56.4.8: SAI clock generator shall be used with FRL = 255, in order to generate the proper frame rate ($F_{FS,x}$).
56.4.12 SPDIF output

The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958. To select SPDIF mode, set PRTCFG[1:0] bit to 01 in the SAI_xCR1 register.

For SPDIF protocol:
- Only SD data line is enabled.
- FS, SCK, MCLK I/Os pins are left free.
- MODE[1] bit is forced to 0 to select the master mode in order to enable the clock generator of the SAI and manage the data rate on the SD line.
- The data size is forced to 24 bits. The value set in DS[2:0] bits in the SAI_xCR1 register is ignored.
- The clock generator must be configured to define the symbol-rate, knowing that the bit clock should be twice the symbol-rate. The data is coded in Manchester protocol.
- The SAI_xFRCR and SAI_xSLOTR registers are ignored. The SAI is configured internally to match the SPDIF protocol requirements as shown in Figure 626.

A SPDIF block contains 192 frames. Each frame is composed of two 32-bit sub-frames, generally one for the left channel and one for the right channel. Each sub-frame is composed of a SOPD pattern (4-bit) to specify if the sub-frame is the start of a block (and so is identifying a channel A) or if it is identifying a channel A somewhere in the block, or if it is referring to channel B (see Table 410). The next 28 bits of channel information are composed of 24 bits data + 4 status bits.

Figure 626. SPDIF format
The data stored in SAI_xDR has to be filled as follows:
- SAI_xDR[26:24] contain the Channel status, User and Validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, then data shall be mapped on SAI_xDR[23:4].
If the data size is 16 bits, then data shall be mapped on SAI_xDR[23:8].
SAI_xDR[23] always represents the MSB.

**Table 410. SOPD pattern**

<table>
<thead>
<tr>
<th>SOPD</th>
<th>Preamble coding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>last bit is 0</td>
<td>last bit is 1</td>
</tr>
<tr>
<td>B</td>
<td>111101000</td>
<td>00010111            Channel A data at the start of block</td>
</tr>
<tr>
<td>W</td>
<td>11100100</td>
<td>00011011            Channel B data somewhere in the block</td>
</tr>
<tr>
<td>M</td>
<td>11100010</td>
<td>00011101            Channel A data</td>
</tr>
</tbody>
</table>

The transfer is performed always with LSB first.

The SAI first sends the adequate preamble for each sub-frame in a block. The SAI_xDR is then sent on the SD line (manchester coded). The SAI ends the sub-frame by transferring the Parity bit calculated as described in **Table 411**.

**Table 411. Parity bit calculation**

<table>
<thead>
<tr>
<th>SAI_xDR[26:0]</th>
<th>Parity bit P value transferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>odd number of 0</td>
<td>0</td>
</tr>
<tr>
<td>odd number of 1</td>
<td>1</td>
</tr>
</tbody>
</table>

The underrun is the only error flag available in the SAI_xSR register for SPDIF mode since the SAI can only operate in transmitter mode. As a result, the following sequence should be
executed to recover from an underrun error detected via the underrun interrupt or the underrun status bit:

1. Disable the DMA stream (via the DMA peripheral) if the DMA is used.
2. Disable the SAI and check that the peripheral is physically disabled by polling the SAIEN bit in SAI_xCR1 register.
3. Clear the COVRUNDR flag in the SAI_xCLRFR register.
4. Flush the FIFO by setting the FFLUSH bit in SAI_xCR2.
   The software needs to point to the address of the future data corresponding to a start of new block (data for preamble B). If the DMA is used, the DMA source base address pointer should be updated accordingly.
5. Enable again the DMA stream (DMA peripheral) if the DMA used to manage data transfers according to the new source base address.
6. Enable again the SAI by setting SAIEN bit in SAI_xCR1 register.

**Clock generator programming in SPDIF generator mode**

For the SPDIF generator, the SAI shall provide a bit clock twice faster as the symbol-rate. The table hereafter shows usual examples of symbol rates with respect to the audio sampling rate.

### Table 412. Audio sampling frequency versus symbol rates

<table>
<thead>
<tr>
<th>Audio Sampling Frequencies (FS)</th>
<th>Symbol-rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>44.1 kHz</td>
<td>2.8224 MHz</td>
</tr>
<tr>
<td>48 kHz</td>
<td>3.072 MHz</td>
</tr>
<tr>
<td>96 kHz</td>
<td>6.144 MHz</td>
</tr>
<tr>
<td>192 kHz</td>
<td>12.288 MHz</td>
</tr>
</tbody>
</table>

More generally, the relationship between the audio sampling frequency (FS) and the bit clock rate (FSCK_x) is given by the formula:

\[
FS = \frac{FSCK_x}{128}
\]

The bit clock rate is obtained as follows:

\[
FSCK_x = \frac{FSAI_CK_x}{MCKDIV}
\]

**Note:** The above formulas are valid only if NODIV is set to 1 in SAI_ACR1 register.
56.4.13 Specific features

The SAI interface embeds specific features which can be useful depending on the audio protocol selected. These functions are accessible through specific bits of the SAI_xCR2 register.

Mute mode

The mute mode can be used when the audio subblock is a transmitter or a receiver.

Audio subblock in transmission mode

In transmitter mode, the mute mode can be selected at anytime. The mute mode is active for entire audio frames. The MUTE bit in the SAI_xCR2 register enables the mute mode when it is set during an ongoing frame.

The mute mode bit is strobed only at the end of the frame. If it is set at this time, the mute mode is active at the beginning of the new audio frame and for a complete frame, until the next end of frame. The bit is then strobed to determine if the next frame will still be a mute frame.

If the number of slots set through NBSLOT[3:0] bits in the SAI_xSLOTR register is lower than or equal to 2, it is possible to specify if the value sent in mute mode is 0 or if it is the last value of each slot. The selection is done via MUTEVAL bit in the SAI_xCR2 register.

If the number of slots set in NBSLOT[3:0] bits in the SAI_xSLOTR register is greater than 2, MUTEVAL bit in the SAI_xCR2 is meaningless as 0 values are sent on each bit on each slot.

The FIFO pointers are still incremented in mute mode. This means that data present in the FIFO and for which the mute mode is requested are discarded.

Audio subblock in reception mode

In reception mode, it is possible to detect a mute mode sent from the external transmitter when all the declared and valid slots of the audio frame receive 0 for a given consecutive number of audio frames (MUTECNT[5:0] bits in the SAI_xCR2 register).

When the number of MUTE frames is detected, the MUTEDET flag in the SAI_xSR register is set and an interrupt can be generated if MUTEDETIE bit is set in SAI_xCR2.

The mute frame counter is cleared when the audio subblock is disabled or when a valid slot receives at least one data in an audio frame. The interrupt is generated just once, when the counter reaches the value specified in MUTECNT[5:0] bits. The interrupt event is then reinitialized when the counter is cleared.

Note: The mute mode is not available for SPDIF audio blocks.

Mono/stereo mode

In transmitter mode, the mono mode can be addressed, without any data preprocessing in memory, assuming the number of slots is equal to 2 (NBSLOT[3:0] = 0001 in SAI_xSLOTR). In this case, the access time to and from the FIFO will be reduced by 2 since the data for slot 0 is duplicated into data slot 1.

To enable the mono mode,
1. Set MONO bit to 1 in the SAI_xCR1 register.
2. Set NBSLOT to 1 and SLOTEN to 3 in SAI_xSLOTR.
In reception mode, the MONO bit can be set and is meaningful only if the number of slots is equal to 2 as in transmitter mode. When it is set, only slot 0 data will be stored in the FIFO. The data belonging to slot 1 will be discarded since, in this case, it is supposed to be the same as the previous slot. If the data flow in reception mode is a real stereo audio flow with a distinct and different left and right data, the MONO bit is meaningless. The conversion from the output stereo file to the equivalent mono file is done by software.

**Companding mode**

Telecommunication applications can require to process the data to be transmitted or received using a data companding algorithm.

Depending on the COMP[1:0] bits in the SAI_xCR2 register (used only when Free protocol mode is selected), the application software can choose to process or not the data before sending it on SD serial output line (compression) or to expand the data after the reception on SD serial input line (expansion) as illustrated in Figure 628. The two companding modes supported are the µ-Law and the A-Law log which are a part of the CCITT G.711 recommendation.

The companding standard used in the United States and Japan is the µ-Law. It supports 14 bits of dynamic range (COMP[1:0] = 10 in the SAI_xCR2 register).

The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

Both µ-Law or A-Law companding standard can be computed based on 1’s complement or 2’s complement representation depending on the CPL bit setting in the SAI_xCR2 register.

In µ-Law and A-Law standards, data are coded as 8 bits with MSB alignment. Companded data are always 8-bit wide. For this reason, DS[2:0] bits in the SAI_xCR1 register will be forced to 010 when the SAI audio block is enabled (SAIEN bit = 1 in the SAI_xCR1 register) and when one of these two companding modes selected through the COMP[1:0] bits.

If no companding processing is required, COMP[1:0] bits should be kept clear.
Expansion and compression mode are automatically selected through the SAI_xCR2:

- If the SAI audio block is configured to be a transmitter, and if the COMP[1] bit is set in the SAI_xCR2 register, the compression mode will be applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm will be applied.

Output data line management on an inactive slot

In transmitter mode, it is possible to choose the behavior of the SD line output when an inactive slot is sent on the data line (via TRIS bit).

- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in HI-z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which could result in a short circuit. To ensure a gap between transmissions, if the data is lower than 32-bit, the data can be extended to 32-bit by setting bit SLOTSZ[1:0] = 10 in the SAI_xSLOTR register. The SD output pin will then be tri-stated at the end of the LSB of the active slot (during the padding to 0 phase to extend the data to 32-bit) if the following slot is declared inactive.

In addition, if the number of slots multiplied by the slot size is lower than the frame length, the SD output line will be tri-stated when the padding to 0 is done to complete the audio frame.

*Figure 629* illustrates these behaviors.
When the selected audio protocol uses the FS signal as a start of frame and a channel side identification (bit FSDEF = 1 in the SAI_xFRCR register), the tristate mode is managed according to Figure 630 (where bit TRIS in the SAI_xCR1 register = 1, and FSDEF=1, and half frame length is higher than number of slots/2, and NBSLOT=6).
Serial audio interface (SAI) RM0455

Figure 630. Tristate on output data line in a protocol like I2S

If the TRIS bit in the SAI_xCR2 register is cleared, all the High impedance states on the SD output line on Figure 629 and Figure 630 are replaced by a drive with a value of 0.

56.4.14 Error flags

The SAI implements the following error flags:
- FIFO overrun/underrun
- Anticipated frame synchronization detection
- Late frame synchronization detection
- Codec not ready (AC’97 exclusively)
- Wrong clock configuration in master mode.

FIFO overrun/underrun (OVRUDR)

The FIFO overrun/underrun bit is called OVRUDR in the SAI_xSR register.

The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

Overrun

When the audio block is configured as receiver, an overrun condition may appear if data are received in an audio frame when the FIFO is full and not able to store the received data. In this case, the received data are lost, the flag OVRUDR in the SAI_xSR register is set and an interrupt is generated if OVRUDRIE bit is set in the SAI_xIM register. The slot number, from which the overrun occurs, is stored internally. No more data will be stored into the FIFO until it becomes free to store new data. When the FIFO has at least one data free, the SAI audio block receiver will store new data (from new audio frame) from the slot number which was stored internally when the overrun condition was detected. This avoids data slot de-alignment in the destination memory (refer to Figure 631).
The OVRUDR flag is cleared when COVRUDR bit is set in the SAI_xCLRFR register.

Figure 631. Overrun detection error

![Diagram showing overrun detection error]

Underrun

An underrun may occur when the audio block in the SAI is a transmitter and the FIFO is empty when data need to be transmitted. If an underrun is detected, the slot number for which the event occurs is stored and MUTE value (00) is sent until the FIFO is ready to transmit the data corresponding to the slot for which the underrun was detected (refer to Figure 632). This avoids desynchronization between the memory pointer and the slot in the audio frame.

The underrun event sets the OVRUDR flag in the SAI_xSR register and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. To clear this flag, set COVRUDR bit in the SAI_xCLRFR register.

The underrun event can occur when the audio subblock is configured as master or slave.

Figure 632. FIFO underrun event

![Diagram showing FIFO underrun event]
Anticipated frame synchronization detection (AFSDET)

The AFSDET flag is used only in slave mode. It is never asserted in master mode. It indicates that a frame synchronization (FS) has been detected earlier than expected since the frame length, the frame polarity, the frame offset are defined and known.

Anticipated frame detection sets the AFSDET flag in the SAI_xSR register.

This detection has no effect on the current audio frame which is not sensitive to the anticipated FS. This means that “parasitic” events on signal FS are flagged without any perturbation of the current audio frame.

An interrupt is generated if the AFSDETIE bit is set in the SAI_xIM register. To clear the AFSDET flag, CAFSDET bit must be set in the SAI_xCLRFR register.

To resynchronize with the master after an anticipated frame detection error, four steps are required:

1. Disable the SAI block by resetting SAIEN bit in SAI_xCR1 register. To make sure the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via FFLUS bit in SAI_xCR2 register.
3. Enable again the SAI peripheral (SAIEN bit set to 1).
4. The SAI block will wait for the assertion on FS to restart the synchronization with master.

Note: The AFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave. It has no meaning in SPDIF mode since the FS signal is not used.

Late frame synchronization detection

The LFSDET flag in the SAI_xSR register can be set only when the SAI audio block operates as a slave. The frame length, the frame polarity and the frame offset configuration are known in register SAI_xFRCR.

If the external master does not send the FS signal at the expecting time thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if LFSDETIE bit is set in the SAI_xIM register.

The LFSDET flag is cleared when CLFSDET bit is set in the SAI_xCLRFR register.

The late frame synchronization detection flag is set when the corresponding error is detected. The SAI needs to be resynchronized with the master (see sequence described in Anticipated frame synchronization detection (AFSDET)).

In a noisy environment, glitches on the SCK clock may be wrongly detected by the audio block state machine and shift the SAI data at a wrong frame position. This event can be detected by the SAI and reported as a late frame synchronization detection error.

There is no corruption if the external master is not managing the audio data frame transfer in continuous mode, which should not be the case in most applications. In this case, the LFSDET flag will be set.

Note: The LFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave. It has no meaning in SPDIF mode since the signal FS is not used by the protocol.
Codec not ready (CNRDY AC’97)

The CNRDY flag in the SAI_xSR register is relevant only if the SAI audio block is configured to operate in AC’97 mode (PRTCFG[1:0] = 10 in the SAI_xCR1 register). If CNRDYIE bit is set in the SAI_xIM register, an interrupt is generated when the CNRDY flag is set.

CNRDY is asserted when the Codec is not ready to communicate during the reception of the TAG 0 (slot0) of the AC’97 audio frame. In this case, no data will be automatically stored into the FIFO since the Codec is not ready, until the TAG 0 indicates that the Codec is ready. All the active slots defined in the SAI_xSLOTR register will be captured when the Codec is ready.

To clear CNRDY flag, CCNRDY bit must be set in the SAI_xCLRFR register.

Wrong clock configuration in master mode (with NODIV = 0)

When the audio block operates as a master (MODE[1] = 0) and NODIV bit is equal to 0, the WCKCFG flag is set as soon as the SAI is enabled if the following conditions are met:
- (FRL+1) is not a power of 2, and
- (FRL+1) is not between 8 and 256.

MODE, NODIV, and SAIEN bits belong to SAI_xCR1 register and FRL to SAI_xFRCR register.

If WCKCFGIE bit is set, an interrupt is generated when WCKCFG flag is set in the SAI_xSR register. To clear this flag, set CWCKCFG bit in the SAI_xCLRFR register.

When WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of SAIEN bit.

56.4.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing SAIEN bit in the SAI_xCR1 register. All the already started frames are automatically completed before the SAI is stops working. SAIEN bit remains High until the SAI is completely switched-off at the end of the current audio frame transfer.

If an audio block in the SAI operates synchronously with the other one, the one which is the master must be disabled first.

56.4.16 SAI DMA interface

To free the CPU and to optimize bus bandwidth, each SAI audio block has an independent DMA interface to read/write from/to the SAI_xDR register (to access the internal FIFO). There is one DMA channel per audio subblock supporting basic DMA request/acknowledge protocol.

To configure the audio subblock for DMA transfer, set DMAEN bit in the SAI_xCR1 register. The DMA request is managed directly by the FIFO controller depending on the FIFO threshold level (for more details refer to Section 56.4.9: Internal FIFOs). DMA transfer direction is linked to the SAI audio subblock configuration:
- If the audio block operates as a transmitter, the audio block FIFO controller outputs a DMA request to load the FIFO with data written in the SAI_xDR register.
- If the audio block is operates as a receiver, the DMA request is related to read operations from the SAI_xDR register.
Follow the sequence below to configure the SAI interface in DMA mode:
1. Configure SAI and FIFO threshold levels to specify when the DMA request will be launched.
2. Configure SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

Note: Before configuring the SAI block, the SAI DMA channel must be disabled.

56.5 SAI interrupts

The SAI supports 7 interrupt sources as shown in Table 413.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt source</th>
<th>Interrupt group</th>
<th>Audio block mode</th>
<th>Interrupt enable</th>
<th>Interrupt clear</th>
</tr>
</thead>
</table>
| FREQ              | FREQ             | FREQ            | Master or slave Receiver or transmitter | FREQIE in SAI_xIM register | Depends on:  
- FIFO threshold setting (FLVL bits in SAI_xCR2)  
- Communication direction (transmitter or receiver)  
For more details refer to Section 56.4.9: Internal FIFOs |
| OVRUDR            | ERROR            | Master or slave Receiver or transmitter | OVRUDRIE in SAI_xIM register | COVRUDR = 1 in SAI_xCLRFR register |
| AFSDET            | ERROR            | Slave (not used in AC'97 mode and SPDIF mode) | AFSDETIE in SAI_xIM register | CAFSDET = 1 in SAI_xCLRFR register |
| LFSDET            | ERROR            | Slave (not used in AC'97 mode and SPDIF mode) | LFSDETIE in SAI_xIM register | CLFSDET = 1 in SAI_xCLRFR register |
| CNRDY             | ERROR            | Slave (only in AC'97 mode) | CNRDIYE in SAI_xIM register | CCNRDY = 1 in SAI_xCLRFR register |
| MUTEDET           | MUTE             | Master or slave Receiver mode only | MUTEDETIE in SAI_xIM register | CMUTEDET = 1 in SAI_xCLRFR register |
| WCKCFG            | ERROR            | Master with NODIV = 0 in SAI_xCR1 register | WCKCFGIE in SAI_xIM register | CWCKCFG = 1 in SAI_xCLRFR register |
Follow the sequence below to enable an interrupt:
1. Disable SAI interrupt.
2. Configure SAI.
3. Configure SAI interrupt source.
4. Enable SAI.
### 56.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

#### 56.6.1 SAI global configuration register (SAI_GCR)

Address offset: 0x00

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-6</th>
<th>Bit 5-4</th>
<th>Bit 3-2</th>
<th>Bit 1-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>MCKDIV[5:0]</td>
<td>NODIV</td>
<td>Res.</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **SYNCOUT[1:0]**: Synchronization outputs

These bits are set and cleared by software.

- **00**: No synchronization output signals. **SYNCOUT[1:0]** should be configured as "No synchronization output signals" when audio block is configured as SPDIF
- **01**: Block A used for further synchronization for others SAI
- **10**: Block B used for further synchronization for others SAI
- **11**: Reserved. These bits must be set when both audio block (A and B) are disabled.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **SYNCIN[1:0]**: Synchronization inputs

These bits are set and cleared by software.

Refer to [Table 405: External synchronization selection](#) for information on how to program this field.

These bits must be set when both audio blocks (A and B) are disabled.

They are meaningful if one of the two audio blocks is defined to operate in synchronous mode with an external SAI (SYNCEN[1:0] = 10 in SAI_ACR1 or in SAI_BCR1 registers).

#### 56.6.2 SAI configuration register 1 (SAI_ACR1)

Address offset: 0x004

Reset value: 0x0000 0040

<table>
<thead>
<tr>
<th>Bit 31-6</th>
<th>Bit 5-4</th>
<th>Bit 3-2</th>
<th>Bit 1-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**OUTD RIV**

**MONO**

**SYNCN[1:0]**

**CKSTR**

**LSBFIRST**

**DS[2:0]**

**PRTCFG[1:0]**

**MODE[1:0]**
Bits 31:28  Reserved, must be kept at reset value.

Bit 27  **MCKEN**: Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26  **OSR**: Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = \( F_{FS} \times 256 \)
1: Master clock frequency = \( F_{FS} \times 512 \)

Bits 25:20  **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, The master clock frequency is calculated according to the formula given in Section 56.4.8: SAI clock generator.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19  **NODIV**: No divider
This bit is set and cleared by software.
0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18  Reserved, must be kept at reset value.

Bit 17  **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled

*Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.*

Bit 16  **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command will not be taken into account.
This bit allows controlling the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.

*Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.*

Bits 15:14  Reserved, must be kept at reset value.

Bit 13  **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.

*Note: This bit has to be set before enabling the audio block and after the audio block configuration.*
Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2.
When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to Section : Mono/stereo mode for more details.
0: Stereo mode
1: Mono mode.

Bits 11:10 **SYNCEN[1:0]**: Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.
00: audio subblock in asynchronous mode.
01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode
10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock should be configured in Slave mode.
11: Reserved

*Note: The audio subblock should be configured as asynchronous when SPDIF mode is enabled.*

Bit 9 **CKSTR**: Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.
0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.
1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST**: Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC’97 audio protocol since AC’97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.
0: Data are transferred with MSB first
1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]**: Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.
These bits must be configured when the audio block is disabled.
000: Reserved
001: Reserved
010: 8 bits
011: 10 bits
100: 16 bits
101: 20 bits
110: 24 bits
111: 32 bits
Bit 4  Reserved, must be kept at reset value.

Bits 3:2  **PRTCFG[1:0]: Protocol configuration**

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol allows to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC’97 protocol

11: Reserved

Bits 1:0  **MODE[1:0]: SAIx audio block mode**

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

*Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00).*

### 56.6.3  SAI configuration register 1 (SAI_BCR1)

**Address offset:** 0x024

**Reset value:** 0x0000 0040

| Bit 31-28 | Reserved, must be kept at reset value. |
| Bit 27   | **MCKEN:** Master clock generation enable |
|          | 0: The master clock is not generated |
|          | 1: The master clock is generated independently of SAIEN bit |
| Bit 26   | **OSR:** Oversampling ratio for master clock |
|          | This bit is meaningful only when NODIV bit is set to 0. |
|          | 0: Master clock frequency = $F_{FS} \times 256$ |
|          | 1: Master clock frequency = $F_{FS} \times 512$ |
| Bits 25:20 | **MCKDIV[5:0]: Master clock divider** |
|          | These bits are set and cleared by software. |
|          | 000000: Divides by 1 the kernel clock input ($sai_x_ker_ck$). |
|          | Otherwise, The master clock frequency is calculated according to the formula given in Section 56.4.8: SAI clock generator. |
|          | These bits have no meaning when the audio block is slave. |
|          | They have to be configured when the audio block is disabled. |
Bit 19 **NODIV**: No divider
   This bit is set and cleared by software.
   0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
   1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
   This bit is set and cleared by software.
   0: DMA disabled
   1: DMA enabled
   **Note:** Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable
   This bit is set by software.
   To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command will not be taken into account.
   This bit allows controlling the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
   0: SAI audio block disabled
   1: SAI audio block enabled.
   **Note:** When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive
   This bit is set and cleared by software.
   0: Audio block output driven when SAIEN is set
   1: Audio block output driven immediately after the setting of this bit.
   **Note:** This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode
   This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to Section : Mono/stereo mode for more details.
   0: Stereo mode
   1: Mono mode.

Bits 11:10 **SYNCEN[1:0]**: Synchronization enable
   These bits are set and cleared by software. They must be configured when the audio subblock is disabled.
   00: audio subblock in asynchronous mode.
   01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode.
   10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock should be configured in Slave mode.
   11: Reserved
   **Note:** The audio subblock should be configured as asynchronous when SPDIF mode is enabled.
Bit 9 **CKSTR:** Clock strobing edge
   This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.
   0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.
   1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first
   This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC’97 audio protocol since AC’97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.
   0: Data are transferred with MSB first
   1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size
   These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.
   These bits must be configured when the audio block is disabled.
   000: Reserved
   001: Reserved
   010: 8 bits
   011: 10 bits
   100: 16 bits
   101: 20 bits
   110: 24 bits
   111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration
   These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.
   00: Free protocol. Free protocol allows to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.
   01: SPDIF protocol
   10: AC’97 protocol
   11: Reserved

Bits 1:0 **MODE[1:0]:** SAI\textsubscript{x} audio block mode
   These bits are set and cleared by software. They must be configured when SAI\textsubscript{x} audio block is disabled.
   00: Master transmitter
   01: Master receiver
   10: Slave transmitter
   11: Slave receiver

*Note:* When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00). In Master transmitter mode, the audio block starts generating the FS and the clocks immediately.
56.6.4      SAI configuration register 2 (SAI_ACR2)

Address offset: 0x008
Reset value: 0x0000 0000

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

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:14  **COMP[1:0]:** Companding mode.

These bits are set and cleared by software. The μ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that will be used depends on CPL bit.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to Section: **Companding mode** for more details.

00: No companding algorithm
01: Reserved.
10: μ-Law algorithm
11: A-Law algorithm

**Note:** Companding mode is applicable only when Free protocol mode is selected.

Bit 13  **CPL:** Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1's complement representation.
1: 2's complement representation.

**Note:** This bit has effect only when the companding mode is μ-Law algorithm or A-Law algorithm.

Bits 12:7  **MUTECNT[5:0]:** Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET will be set and an interrupt will be generated if bit MUTEDETIE is set.

Refer to Section: **Mute mode** for more details.
Bit 6 **MUTEVAL**: Mute value.
This bit is set and cleared by software. It must be written before enabling the audio block; SAIEN. This bit is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set. If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL. If the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame. Refer to *Section: Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.
1: Last values are sent during the mute mode.

**Note:** This bit is meaningless and should not be used for SPDIF audio blocks.

Bit 5 **MUTE**: Mute.
This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2. Refer to *Section: Mute mode* for more details.

0: No mute mode.
1: Mute mode enabled.

**Note:** This bit is meaningless and should not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.
This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It should be configured when SAI is disabled. Refer to *Section: Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.
1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH**: FIFO flush.
This bit is set by software. It is always read as 0. This bit should be configured when the SAI is disabled.

0: No FIFO flush.
1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled.

Bits 2:0 **FTH[2:0]**: FIFO threshold.
This bit is set and cleared by software.

000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved
### 56.6.5 SAI configuration register 2 (SAI_BCR2)

Address offset: 0x028

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:16 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 15:14 <strong>COMP[1:0]:</strong> Companding mode.</td>
</tr>
<tr>
<td>These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that will be used depends on CPL bit.</td>
</tr>
<tr>
<td>The data expansion or data compression are determined by the state of bit MODE[0].</td>
</tr>
<tr>
<td>The data compression is applied if the audio block is configured as a transmitter.</td>
</tr>
<tr>
<td>The data expansion is automatically applied when the audio block is configured as a receiver.</td>
</tr>
<tr>
<td>Refer to <strong>Section : Companding mode</strong> for more details.</td>
</tr>
<tr>
<td>00: No companding algorithm</td>
</tr>
<tr>
<td>01: Reserved.</td>
</tr>
<tr>
<td>10: µ-Law algorithm</td>
</tr>
<tr>
<td>11: A-Law algorithm</td>
</tr>
<tr>
<td><strong>Note:</strong> Companding mode is applicable only when Free protocol mode is selected.</td>
</tr>
<tr>
<td>Bit 13 <strong>CPL:</strong> Complement bit.</td>
</tr>
<tr>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td>It defines the type of complement to be used for companding mode</td>
</tr>
<tr>
<td>0: 1’s complement representation.</td>
</tr>
<tr>
<td>1: 2’s complement representation.</td>
</tr>
<tr>
<td><strong>Note:</strong> This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.</td>
</tr>
<tr>
<td>Bits 12:7 <strong>MUTECNT[5:0]:</strong> Mute counter.</td>
</tr>
<tr>
<td>These bits are set and cleared by software. They are used only in reception mode.</td>
</tr>
<tr>
<td>The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET will be set and an interrupt will be generated if bit MUTEDETIE is set.</td>
</tr>
<tr>
<td>Refer to <strong>Section : Mute mode</strong> for more details.</td>
</tr>
</tbody>
</table>
Bit 6 **MUTEVAL**: Mute value.
This bit is set and cleared by software. It must be written before enabling the audio block: SAIEN. This bit is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.
If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.
If the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.
Refer to Section : Mute mode for more details.
0: Bit value 0 is sent during the mute mode.
1: Last values are sent during the mute mode.
*Note:* This bit is meaningless and should not be used for SPDIF audio blocks.

Bit 5 **MUTE**: Mute.
This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.
Refer to Section : Mute mode for more details.
0: No mute mode.
1: Mute mode enabled.
*Note:* This bit is meaningless and should not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.
This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It should be configured when SAI is disabled.
Refer to Section : Output data line management on an inactive slot for more details.
0: SD output line is still driven by the SAI when a slot is inactive.
1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH**: FIFO flush.
This bit is set by software. It is always read as 0. This bit should be configured when the SAI is disabled.
0: No FIFO flush.
1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled.

Bits 2:0 **FTH[2:0]**: FIFO threshold.
This bit is set and cleared by software.
000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved
56.6.6  SAI frame configuration register (SAI_AFRCR)

Address offset: 0x00C
Reset value: 0x0000 0007

Note:  This register has no meaning in AC’97 and SPDIF audio protocol.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res.</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>Res.</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>Res.</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>Res.</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>Res.</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>Res.</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>Res.</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>Res.</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>Res.</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>Res.</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>Res.</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>Res.</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>Res.</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>FSOFF: Frame synchronization offset.</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>FSPOL: Frame synchronization polarity.</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>FSDEF: Frame synchronization definition.</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>FSALL[8:0]</td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td>FRL[7:0]</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:19  Reserved, must be kept at reset value.

Bit 18  **FSOFF**: Frame synchronization offset.

This bit is set and cleared by software. It is meaningless and is not used in AC’97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

Bit 17  **FSPOL**: Frame synchronization polarity.

This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC’97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16  **FSDEF**: Frame synchronization definition.

This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTTR register has to be even. It means that half of this number of slots will be dedicated to the left channel and the other slots for the right channel (e.g.: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC’97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.
Bit 15  Reserved, must be kept at reset value.

Bits 14:8  **FSALL[6:0]**: Frame synchronization active level length.
           These bits are set and cleared by software. They specify the length in number of bit clock
           (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame.
           These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration.
           They must be configured when the audio block is disabled.

Bits 7:0  **FRL[7:0]**: Frame length.
           These bits are set and cleared by software. They define the audio frame length expressed in number
           of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.
           The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio
           block will behaves in an unexpected way. This is the case when the data size is 8 bits and only one
           slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).
           In master mode, if the master clock (available on MCLK_x pin) is used, the frame length should be
           aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not
           used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.
           These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration. They
           must be configured when the audio block is disabled.

### 56.6.7  SAI frame configuration register (SAI_BFRCR)

Address offset: 0x02C
Reset value: 0x0000 0007

**Note:**  *This register has no meaning in AC’97 and SPDIF audio protocol*

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>FSALL[6:0]</td>
<td>FRL[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:19  Reserved, must be kept at reset value.

**Bit 18  FSOFF:** Frame synchronization offset.
This bit is set and cleared by software. It is meaningless and is not used in AC’97 or SPDIF audio
block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

**Bit 17  FSPOL:** Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS
signal. It is meaningless and is not used in AC’97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)
Bit 16 **FSDEF**: Frame synchronization definition.

This bit is set and cleared by software.

0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification

When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots will be dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).

This bit is meaningless and is not used in AC’97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.

These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame.

These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration. They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block will behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length should be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256. These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration.

**56.6.8 SAI slot register (SAI_ASLOTR)**

Address offset: 0x010
Reset value: 0x0000 0000

*Note:* This register has no meaning in AC’97 and SPDIF audio protocol.
Bits 31:16 **SLOTEN[15:0]**: Slot enable.
These bits are set and cleared by software.
Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.
These bits are set and cleared by software.
The value set in this bitfield represents the number of slots + 1 in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots should be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI will be undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

---

**56.6.9 SAI slot register (SAI_BSLOTR)**

Address offset: 0x030
Reset value: 0x0000 0000

*Note:* This register has no meaning in AC'97 and SPDIF audio protocol.
56.6.10 SAI interrupt mask register (SAI_AIM)

Address offset: 0x014

Reset value: 0x0000 0000

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

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

- LFSDET IE
- AFSDKETE IE
- CNRDY IE
- FREQ IE
- WCKCFG IE
- MUTEDT IE
- OVRUDR IE

2220/2974
Bits 31:7  Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When this bit is set, an interrupt will be generated if the LFSDET bit is set in the SAI_xSR register.
   This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When this bit is set, an interrupt will be generated if the AFSDET bit in the SAI_xSR register is set.
   This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC’97).
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC’97 frame if the
   Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR
   register is set and an interrupt is generated.
   This bit has a meaning only if the AC’97 mode is selected through PRTCFOG[1:0] bits and the audio
   block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
   Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be
   configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and
   NODIV = 0.
   It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
   Note:  This bit is used only in Free protocol mode and is meaningless in other modes.

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
   This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.
### 56.6.11 SAI interrupt mask register (SAI_BIM)

Address offset: 0x034  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
| 15   | LFSDETIE: Late frame synchronization detection interrupt enable.  
        This bit is set and cleared by software.  
        0: Interrupt is disabled  
        1: Interrupt is enabled  
        When this bit is set, an interrupt will be generated if the LFSDET bit is set in the SAI_xSR register.  
        This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master. |
| 14   | Reserved                                        |
| 13   | Reserved                                        |
| 12   | Reserved                                        |
| 11   | Reserved                                        |
| 10   | Reserved                                        |
| 9    | Reserved                                        |
| 8    | Reserved                                        |
| 7    | Reserved                                        |
| 6    | LFSDETIE: Late frame synchronization detection interrupt enable.  
        This bit is set and cleared by software.  
        0: Interrupt is disabled  
        1: Interrupt is enabled  
        When this bit is set, an interrupt will be generated if the LFSDET bit in the SAI_xSR register is set.  
        This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master. |
| 5    | AFSDETIE: Anticipated frame synchronization detection interrupt enable.  
        This bit is set and cleared by software.  
        0: Interrupt is disabled  
        1: Interrupt is enabled  
        When this bit is set, an interrupt will be generated if the AFSDET bit in the SAI_xSR register is set.  
        This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master. |
| 4    | CNRDYIE: Codec not ready interrupt enable (AC’97).  
        This bit is set and cleared by software.  
        0: Interrupt is disabled  
        1: Interrupt is enabled  
        When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC’97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.  
        This bit has a meaning only if the AC’97 mode is selected through PRTC[1:0] bits and the audio block is operates as a receiver. |
| 3    | FREQIE: FIFO request interrupt enable.  
        This bit is set and cleared by software.  
        0: Interrupt is disabled  
        1: Interrupt is enabled  
        When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.  
        Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode, |

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.
Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
- This bit is set and cleared by software.
- 0: Interrupt is disabled
- 1: Interrupt is enabled

This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.

It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.

*Note: This bit is used only in Free protocol mode and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
- This bit is set and cleared by software.
- 0: Interrupt is disabled
- 1: Interrupt is enabled

When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.

This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
- This bit is set and cleared by software.
- 0: Interrupt is disabled
- 1: Interrupt is enabled

When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

### 56.6.12 SAI status register (SAI_ASR)

**Address offset**: 0x018  
**Reset value**: 0x0000 0008

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.

- This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
- 000: FIFO empty (transmitter and receiver modes)
- 001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
- 010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
- 011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
- 100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
- 101: FIFO full (transmitter and receiver modes)
- Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.
Bit 6 **LFSDET**: Late frame synchronization detection.
   - This bit is read only.
   - 0: No error.
   - 1: Frame synchronization signal is not present at the right time.
   - This flag can be set only if the audio block is configured in slave mode.
   - It is not used in AC’97 or SPDIF mode.
   - It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
   - This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register.

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
   - This bit is read only.
   - 0: No error.
   - 1: Frame synchronization signal is detected earlier than expected.
   - This flag can be set only if the audio block is configured in slave mode.
   - It is not used in AC’97 or SPDIF mode.
   - It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
   - This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
   - This bit is read only.
   - 0: External AC’97 Codec is ready
   - 1: External AC’97 Codec is not ready
   - This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
   - It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
   - This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
   - This bit is read only.
   - 0: No FIFO request.
   - 1: FIFO request to read or to write the SAI_xDR.
   - The request depends on the audio block configuration:
     - If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
     - If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
   - This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.
Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in
Section 56.4.6: Frame synchronization (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio
frame
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a
consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a
receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

56.6.13 **SAI status register (SAI_BSR)**
Address offset: 0x038
Reset value: 0x0000 0008

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>
Bits 31:19  Reserved, must be kept at reset value.

Bits 18:16  **FLVL[2:0]**: FIFO level threshold.

- This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
- 000: FIFO empty (transmitter and receiver modes)
- 001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
- 010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
- 011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
- 100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
- 101: FIFO full (transmitter and receiver modes)
- Others: Reserved

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

Bit 6  **LFSDET**: Late frame synchronization detection.

- This bit is read only.
- 0: No error.
- 1: Frame synchronization signal is not present at the right time.
- This flag can be set only if the audio block is configured in slave mode.
- It is not used in AC’97 or SPDIF mode.
- It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
- This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register.

Bit 5  **AFSDET**: Anticipated frame synchronization detection.

- This bit is read only.
- 0: No error.
- 1: Frame synchronization signal is detected earlier than expected.
- This flag can be set only if the audio block is configured in slave mode.
- It is not used in AC’97 or SPDIF mode.
- It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
- This flag is cleared when the software sets CAFSDET in SAI_xCLRFR register.

Bit 4  **CNRDY**: Codec not ready.

- This bit is read only.
- 0: External AC’97 Codec is ready
- 1: External AC’97 Codec is not ready
- This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
- It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
- This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3  **FREQ**: FIFO request.

- This bit is read only.
- 0: No FIFO request.
- 1: FIFO request to read or to write the SAI_xDR.
- The request depends on the audio block configuration:
  - If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
  - If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
- This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.
Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in the [Section 56.4.6: Frame synchronization](#)**. This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0. It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frames
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

### 56.6.14 SAI clear flag register (SAI_ACLRFR)

Address offset: 0x01C
Reset value: 0x0000 0000

- **Res.**: Read-only
- **Res.**: Readable/writeable

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

Address offset: 0x03C
Reset value: 0x0000 0000

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 CLFSDET: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 5 CAFSDET: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 CCNRDY: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 CWCKCFG: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 CMUTEDET: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 COVRUDR: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

---

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

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

---

56.6.16 SAI data register (SAI_ADR)

Address offset: 0x020
Reset value: 0x0000 0000

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

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 CLFSDET: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 5 CAFSDDET: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 CCNRDY: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 CWCKCFG: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 CMUTEDET: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 COVRUDR: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.
### 56.6.17 SAI data register (SAI_BDR)

**Address offset:** 0x040  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:0</th>
<th>Description</th>
<th>Access</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA[31:0]</td>
<td>Data</td>
<td>rw</td>
<td>0x0000 0000</td>
</tr>
</tbody>
</table>

- A write to this register loads the FIFO provided the FIFO is not full.  
- A read from this register empties the FIFO if the FIFO is not empty.

**Bits 15:0**  
DATA[15:0]  
<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA[15:0]</td>
<td>Data</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bits 31:16 Reserved, must be kept at reset value.**

**Bits 15:12 Reserved, must be kept at reset value.**

**Bit 11 CKEN4:** Clock enable of bitstream clock number 4  
- This bit is set and cleared by software.  
- 0: SAI_CK4 clock disabled  
- 1: SAI_CK4 clock enabled  

**Note:** It is not recommended to configure this bit when PDMEN = 1.
Bit 10 **CKEN3**: Clock enable of bitstream clock number 3
   This bit is set and cleared by software.
   0: SAI_CK3 clock disabled
   1: SAI_CK3 clock enabled

   Note: It is not recommended to configure this bit when PDMEN = 1.

Bit 9 **CKEN2**: Clock enable of bitstream clock number 2
   This bit is set and cleared by software.
   0: SAI_CK2 clock disabled
   1: SAI_CK2 clock enabled

   Note: It is not recommended to configure this bit when PDMEN = 1.

Bit 8 **CKEN1**: Clock enable of bitstream clock number 1
   This bit is set and cleared by software.
   0: SAI_CK1 clock disabled
   1: SAI_CK1 clock enabled

   Note: It is not recommended to configure this bit when PDMEN = 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MICNBR[1:0]**: Number of microphones
   This bit is set and cleared by software.
   00: Configuration with 2 microphones
   01: Configuration with 4 microphones
   10: Configuration with 6 microphones
   11: Configuration with 8 microphones

   Note: It is not recommended to configure this field when PDMEN = 1.*

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PDMEN**: PDM enable
   This bit is set and cleared by software. This bit allows to control the state of the PDM interface block.
   Make sure that the SAI is already operating in TDM master mode before enabling the PDM interface.
   0: PDM interface disabled
   1: PDM interface enabled

### 56.6.19 SAI PDM delay register (SAI_PDMDLY)

Address offset: 0x0048
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>DLYM4R[2:0]</td>
</tr>
<tr>
<td>30</td>
<td>DLYM4L[2:0]</td>
</tr>
<tr>
<td>29</td>
<td>DLYM3R[2:0]</td>
</tr>
<tr>
<td>28</td>
<td>DLYM3L[2:0]</td>
</tr>
<tr>
<td>27</td>
<td>DLYM2R[2:0]</td>
</tr>
<tr>
<td>26</td>
<td>DLYM2L[2:0]</td>
</tr>
<tr>
<td>25</td>
<td>DLYM1R[2:0]</td>
</tr>
<tr>
<td>24</td>
<td>DLYM1L[2:0]</td>
</tr>
</tbody>
</table>

Note: 000: Configuration with 2 microphones
      010: Configuration with 4 microphones
      100: Configuration with 6 microphones
      110: Configuration with 8 microphones

Note: It is not recommended to configure this field when PDMEN = 1.*
Bit 31  Reserved, must be kept at reset value.

Bits 30:28  **DLYM4R[2:0]**: Delay line for second microphone of pair 4
This bit is set and cleared by software.
000: No delay
001: Delay of 1 $T_{\text{SAI\_CK}}$ period
010: Delay of 2 $T_{\text{SAI\_CK}}$ periods
... 111: Delay of 7 $T_{\text{SAI\_CK}}$ periods
This field can be changed on-the-fly.

Bit 27  Reserved, must be kept at reset value.

Bits 26:24  **DLYM4L[2:0]**: Delay line for first microphone of pair 4
This bit is set and cleared by software.
000: No delay
001: Delay of 1 $T_{\text{SAI\_CK}}$ period
010: Delay of 2 $T_{\text{SAI\_CK}}$ periods
... 111: Delay of 7 $T_{\text{SAI\_CK}}$ periods
This field can be changed on-the-fly.

Bit 23  Reserved, must be kept at reset value.

Bits 22:20  **DLYM3R[2:0]**: Delay line for second microphone of pair 3
This bit is set and cleared by software.
000: No delay
001: Delay of 1 $T_{\text{SAI\_CK}}$ period
010: Delay of 2 $T_{\text{SAI\_CK}}$ periods
... 111: Delay of 7 $T_{\text{SAI\_CK}}$ periods
This field can be changed on-the-fly.

Bit 19  Reserved, must be kept at reset value.

Bits 18:16  **DLYM3L[2:0]**: Delay line for first microphone of pair 3
This bit is set and cleared by software.
000: No delay
001: Delay of 1 $T_{\text{SAI\_CK}}$ period
010: Delay of 2 $T_{\text{SAI\_CK}}$ periods
... 111: Delay of 7 $T_{\text{SAI\_CK}}$ periods
This field can be changed on-the-fly.

Bit 15  Reserved, must be kept at reset value.
Bits 14:12 **DLYM2R[2:0]**: Delay line for second microphone of pair 2
- This bit is set and cleared by software.
- 000: No delay
- 001: Delay of 1 T_{SAL\_CK} period
- 010: Delay of 2 T_{SAL\_CK} periods
- ...
- 111: Delay of 7 T_{SAL\_CK} periods

This field can be changed on-the-fly.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DLYM2L[2:0]**: Delay line for first microphone of pair 2
- This bit is set and cleared by software.
- 000: No delay
- 001: Delay of 1 T_{SAL\_CK} period
- 010: Delay of 2 T_{SAL\_CK} periods
- ...
- 111: Delay of 7 T_{SAL\_CK} periods

This field can be changed on-the-fly.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DLYM1R[2:0]**: Delay line adjust for second microphone of pair 1
- This bit is set and cleared by software.
- 000: No delay
- 001: Delay of 1 T_{SAL\_CK} period
- 010: Delay of 2 T_{SAL\_CK} periods
- ...
- 111: Delay of 7 T_{SAL\_CK} periods

This field can be changed on-the-fly.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DLYM1L[2:0]**: Delay line adjust for first microphone of pair 1
- This bit is set and cleared by software.
- 000: No delay
- 001: Delay of 1 T_{SAL\_CK} period
- 010: Delay of 2 T_{SAL\_CK} periods
- ...
- 111: Delay of 7 T_{SAL\_CK} periods

This field can be changed on-the-fly.
## 56.6.20 SAI register map

The following table summarizes the SAI registers.

| Offset       | Register name | Offset Name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------------|---------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x0000       | SAI_GCR       |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0004 or   | SAI_xCR1      |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0024       |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0008 or   | SAI_xCR2      |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0028       |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x000C or   | SAI_xFRCR     |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x002C       |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0010 or   | SAI_xSLOTTR   | SLOTEN[15:0]|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0030       |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0014 or   | SAI_xIM       |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0034       |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0018 or   | SAI_xSR       |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0038       |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x001C or   | SAI_xCLRFR    |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x003C       |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0020 or   | SAI_xDR       | DATA[31:0]  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0040       |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Table 414. SAI register map and reset values

- **Reset value**

For each register, the reset value is provided, which specifies the default state of the register upon reset. This includes the initial values for various bits and fields within each register, ensuring that the SAI module starts in a predictable state.
Refer to *Section 2.3 on page 128* for the register boundary addresses.
57 SPDIF receiver interface (SPDIFRX)

57.1 SPDIFRX interface introduction

The SPDIFRX interface handles S/PDIF audio protocol.

57.2 SPDIFRX main features

- Up to 4 inputs available
- Automatic symbol rate detection
- Maximum symbol rate: 12.288 MHz
- Stereo stream from 8 to 192 kHz supported
- Supports Audio IEC-60958 and IEC-61937, consumer applications
- Parity bit management
- Communication using DMA for audio samples
- Communication using DMA for control and user channel information
- Interrupt capabilities

57.3 SPDIFRX functional description

The SPDIFRX peripheral is designed to receive an S/PDIF flow compliant with IEC-60958 and IEC-61937. These standards support simple stereo streams up to high sample rate, and compressed multi-channel surround sound, such as those defined by Dolby or DTS.

The receiver provides all the necessary features to detect the symbol rate, and decode the incoming data. It is possible to use a dedicated path for the user and channel information in order to ease the interface handling. Figure 633 shows a simplified block diagram.

The SPDIFRX_DC block is responsible of the decoding of the S/PDIF stream received from SPDIFRX_IN[4:1] inputs. This block re-sample the incoming signal, decode the manchester stream, recognize frames, sub-frames and blocks elements. It delivers to the REG_IF part, decoded data, and associated status flags.

This peripheral can be fully controlled via the APB1 bus, and can handle two DMA channels:

- A DMA channel dedicated to the transfer of audio samples
- A DMA channel dedicated to the transfer of IEC60958 channel status and user information

Interrupt services are also available either as an alternative function to the DMA, or for signaling error or key status of the peripheral.

The SPDIFRX also offers a signal named spdifrx_frame_sync, which toggles every time that a sub-frame’s preamble is detected. So the duty cycle is 50%, and the frequency equal to the frame rate.

This signal can be connected to timer events, in order to compute frequency drift.

In addition the SPDIFRX also provides a signal named spdifrx_symb_ck toggling at the symbol rate.
1. ‘n’ is fixed to 4.

57.3.1 SPDIFRX pins and internal signals

Table 415 lists the SPDIFRX internal input/output signals, Table 416 the SPDIFRX pins (alternate functions).

Table 415. SPDIFRX internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spdifrx_ker_ck</td>
<td>Digital input</td>
<td>SPDIFRX kernel clock</td>
</tr>
<tr>
<td>spdifrx_pclk</td>
<td>Digital input</td>
<td>SPDIFRX register interface clock</td>
</tr>
<tr>
<td>spdifrx_it</td>
<td>Digital output</td>
<td>SPDIFRX global interrupt</td>
</tr>
<tr>
<td>spdifrx_dat_dma</td>
<td>Digital input/output</td>
<td>SPDIFRX DMA request (and acknowledge) for data transfer</td>
</tr>
<tr>
<td>spdifrx_ctrl_dma</td>
<td>Digital input/output</td>
<td>SPDIFRX DMA request (and acknowledge) for channel status and user information transfer</td>
</tr>
<tr>
<td>spdifrx_frame_sync</td>
<td>Digital output</td>
<td>SPDIFRX frame rate synchronization signal</td>
</tr>
<tr>
<td>spdifrx_symb_ck</td>
<td>Digital output</td>
<td>SPDIFRX channel symbol clock</td>
</tr>
</tbody>
</table>

Table 416. SPDIFRX pins

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPDIFRX_IN1</td>
<td>Digital input</td>
<td>Input 1 for S/PDIF signal</td>
</tr>
<tr>
<td>SPDIFRX_IN2</td>
<td>Digital input</td>
<td>Input 2 for S/PDIF signal</td>
</tr>
</tbody>
</table>
57.3.2 S/PDIF protocol (IEC-60958)

S/PDIF block
A S/PDIF frame is composed of two sub-frames (see Figure 635). Each sub-frame contains 32 bits (or time slots):

- Bits 0 to 3 carry one of the synchronization preambles
- Bits 4 to 27 carry the audio sample word in linear 2’s complement representation. The most significant bit (MSB) is carried by bit 27. When a 20-bit coding range is used, bits 8 to 27 carry the audio sample word with the LSB in bit 8.
- Bit 28 (validity bit “V”) indicates if the data is valid (for converting it to analog for example)
- Bit 29 (user data bit “U”) carries the user data information like the number of tracks of a Compact Disk.
- Bit 30 (channel status bit “C”) carries the channel status information like sample rate and protection against copy.
- Bit 31 (parity bit “P”) carries a parity bit such that bits 4 to 31 inclusive carry an even number of ones and an even number of zeroes (even parity).

For linear coded audio applications, the first sub-frame (left or “A” channel in stereophonic operation and primary channel in monophonic operation) normally starts with preamble “M”. However, the preamble changes to preamble “B” once every 192 frames to identify the start of the block structure used to organize the channel status and user information. The second sub-frame (right or “B” channel in stereophonic operation and secondary channel in monophonic operation) always starts with preamble “W”.

A S/PDIF block contains 192 pairs of sub-frames of 32 bits.
Synchronization preambles

The preambles patterns are inverted or not according to the previous half-bit value. This previous half-bit value is the level of the line before enabling a transfer for the first “B” preamble of the first frame. For the others preambles, this previous half-bit value is the second half-bit of the parity bit of the previous sub-frame. The preambles patterns B, M and W are described in the Figure 636.

Figure 636. S/PDIF Preambles

Coding of information bits

In order to minimize the DC component value on the transmission line, and to facilitate clock recovery from the data stream, bits 4 to 31 are encoded in biphase-mark.

Each bit to be transmitted is represented by a symbol comprising two consecutive binary states. The first state of a symbol is always different from the second state of the previous symbol. The second state of the symbol is identical to the first if the bit to be transmitted is
logical 0. However, it is different if the bit is logical 1. These states are named “UI” (Unit Interval) in the IEC-60958 specification.

The 24 data bits are transferred LSB first.

**Figure 637. Channel coding example**

<table>
<thead>
<tr>
<th>Bit Clock</th>
<th>Source coding</th>
<th>Channel coding (Biphase-Mark)</th>
<th>BitStream Biphase-Mark Coded</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 UI</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2 UI</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3 UI</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4 UI</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 UI</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 UI</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 UI</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 UI</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 UI</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 UI</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 UI</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 UI</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 UI</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 UI</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**57.3.3 SPDIFRX decoder (SPDIFRX_DC)**

**Main principle**

The technique used by the SPDIFRX in order to decode the S/PDIF stream is based on the measurement of the time interval between two consecutive edges. Three kinds of time intervals may be found into an S/PDIF stream:

- The long time interval, having a duration of 3 x UI, noted TL. It appears only during preambles.
- The medium time interval, having a duration of 2 x UI, noted TM. It appears both in some preambles or into the information field.
- The short time interval, having a duration of 1 x UI, noted TS. It appears both in some preambles or into the information field.

The SPDIFRX_DC block is responsible of the decoding of the received S/PDIF stream. It takes care of the following functions:

- Resampling and filtering of the incoming signal
- Estimation of the time-intervals
- Estimation of the symbol rate and synchronization
- Decoding of the serial data, and check of integrity
- Detection of the block, and sub-frame preambles
- Continuous tracking of the symbol rate
Figure 638 gives a detailed view of the SPDIFRX decoder.

**Figure 638. SPDIFRX decoder**

### Noise filtering & rising/falling edge detection

The S/PDIF signal received on the selected SPDIFRX_IN is re-sampled using the spdifrx_ker_ck clock (acquisition clock). A simple filtering is applied in order cancel spurs. This is performed by the stage detecting the edge transitions. The edge transitions are detected as follow:

- A rising edge is detected when the sequence 0 followed by two 1 is sampled.
- A falling edge is detected when the sequence 1 followed by two 0 is sampled.
- After a rising edge, a falling edge sequence is expected.
- After a falling edge, a rising edge sequence is expected.

**Figure 639. Noise filtering and edge detection**

### Longest and shortest transition detector

The **longest and shortest transition detector** block detects the maximum (MAX_CNT) and minimum (MIN_CNT) duration between two transitions. The TRCNT counter is used to measure the time interval duration. It is clocked by the spdifrx_ker_ck signal. On every transition pulse, the counter value is stored and the counter is reset to start counting again.

The maximum duration is normally found during the preamble period. This maximum duration is sent out as MAX_CNT. The minimum duration is sent out as MIN_CNT.
The search of the longest and shortest transition is stopped when the transition timer expires. The transition timer is like a watchdog timer that generates a trigger after 70 transitions of the incoming signal. Note that counting 70 transitions insures a delay a bit longer than a sub-frame.

Note that when the TRCNT overflows due to a too long time interval between two pulses, the SPDIFRX is stopped and the flag TERR of SPDIFRX_SR register is set to 1.

Transition coder and preamble detector

The transition coder and preamble detector block receives the MAX_CNT and MIN_CNT. It also receives the current transition width from the TRCNT counter (see Figure 638). This block encodes the current transition width by comparing the current transition width with two different thresholds, names TH_HI and TH_LO.

- If the current transition width is less than (TH_LO - 1), then the data received is half part of data bit ‘1’, and is coded as TS.
- If the current transition width is greater than (TH_LO - 1), and less than TH_HI, then the data received is data bit ‘0’, and is coded as TM.
- If the current transition width is greater than TH_HI, then the data received is the long pulse of preambles, and is coded as TL.
- Else an error code is generated (FERR flag is set).

The thresholds TH_HI and TH_LO are elaborated using two different methods.

If the peripheral is doing its initial synchronization (‘coarse synchronization’), then the thresholds are computed as follow:

- TH_LO = MAX_CNT / 2.
- TH_HI = MIN_CNT + MAX_CNT / 2.

Once the ‘coarse synchronization’ is completed, then the SPDIFRX uses a more accurate reference in order to elaborate the thresholds. The SPDIFRX measures the length of 24 symbols (WIDTH24) for defining TH_LO and the length of 40 symbols (WIDTH40) for TH_HI. TH_HI and TH_LO are computed as follow:

- TH_LO = (WIDTH24) / 32
- TH_HI = (WIDTH40) / 32

This second synchronization phase is called the ‘fine synchronization’. Refer to Figure 642 for additional information.

As shown in the figure hereafter, TH_LO is ideally equal to 1.5 UI, and to TH_HI 2.5 UI.
The preamble detector checks four consecutive transitions of a specific sequence to determine if they form the part of preamble. Let us say TRANS0, TRANS1, TRANS2 and TRANS3 represent four consecutive transitions encoded as mentioned above. Table 417 shows the values of these four transitions to form a preamble. Absence of this pattern indicates that these transitions form part of the data in the sub frame and bi-phase decoder decode them.

### Table 417. Transition sequence for preamble

<table>
<thead>
<tr>
<th>Preamble type</th>
<th>Biphasic data pattern</th>
<th>TRANS3</th>
<th>TRANS2</th>
<th>TRANS1</th>
<th>TRANS0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preamble B</td>
<td>11101000</td>
<td>TL</td>
<td>TS</td>
<td>TS</td>
<td>TL</td>
</tr>
<tr>
<td>Preamble M</td>
<td>11100010</td>
<td>TL</td>
<td>TL</td>
<td>TS</td>
<td>TS</td>
</tr>
<tr>
<td>Preamble W</td>
<td>11100100</td>
<td>TL</td>
<td>TM</td>
<td>TS</td>
<td>TM</td>
</tr>
</tbody>
</table>

**Bi-phase decoder**

The Bi-phase decoder decodes the input bi-phase marked data stream using the transition information provided by the transition coder and preamble detector block. It first waits for the preamble detection information. After the preamble detection, it decodes the following transition information:

- If the incoming transition information is TM then it is decoded as a ‘0’.
- Two consecutive TS are decoded as a ‘1’.
- Any other transition sequence generates an error signal (FERR set to 1).

After decoding 28 data bits this way, this module looks for the following preamble data. If the new preamble is not what is expected, then this block generates an error signal (FERR set to 1). Refer to Section 57.3.9: Reception errors, for additional information on error flags.

**Data packing**

This block is responsible of the decoding of the IEC-60958 frames and blocks. It also handles the writing into the RX_BUF or into SPDIFRX_CSR register.
57.3.4 SPDIFRX tolerance to clock deviation

The SPDIFRX tolerance to clock deviation depends on the number of sample clock cycles in one bit slot. The fastest spdifrx_ker_ck is, the more robust the reception is. The ratio between spdifrx_ker_ck frequency and the symbol rate must be at least 11.

Two kinds of phenomenon (at least!) can degrade the reception quality:

- The cycle-to-cycle jitter which reflects the difference of transition length between two consecutive transitions.
- The long term jitter which reflects a cumulative effect of the cycle-to-cycle jitter. It can be seen as a low-frequency symbol modulation.

57.3.5 SPDIFRX synchronization

The synchronization phase starts when setting SPDIFRXEN to 01 or 11. Figure 641 shows the synchronization process.

If the bit WFA of SPDIFRX_CR register is set to 1, then the peripheral must first detect activity on the selected SPDIFRX_IN line before starting the synchronization process. The activity detection is performed by detecting four transitions on the selected SPDIFRX_IN. The peripheral remains in this state until transitions are not detected. This function can be particularly helpful because the IP switches in COARSE SYNC mode only if activity is present on the selected SPDIFRX_IN input, avoiding synchronization errors. See Section 57.4: Programming procedures for additional information.

The user can still set the SPDIFRX into STATE_IDLE by setting SPDIFRXEN to 0. If the WFA is set to 0, the peripheral starts the coarse synchronization without checking activity.

The next step consists on doing a first estimate of the thresholds (COARSE SYNC), in order to perform the fine synchronization (FINE SYNC). Due to disturbances of the SPDIFRX line, it could happen that the process is not executed first time right. For this purpose, the user can program the number of allowed re-tries (NBTR) before setting SERR error flag. When the SPDIFRX has been able to measure properly the duration of 24 and 40 consecutive symbols then the FINE SYNC is completed, the threshold values are updated, and the flag SYND is set to 1. Refer to Section : Transition coder and preamble detector for additional information.

Two kinds of errors are detected:

- An overflow of the TRCNT, which generally means that there is no valid S/PDIF stream in the input line. This overflow is indicated by TERR flag.
- The number of retries reached the programmed value. This means that strong jitter is present on the S/PDIF signal. This error is indicated by SERR flag.

When the first FINE SYNC is completed, the reception of channel status (C) and user data (U) starts when the next “B” preamble is detected (see Figure 645). Then the user can read IEC-60958 C and U bits through SPDIFRX_CSR register. According to this information the user can then select the proper settings for DRFMT and RXSTEO. For example if the user detects that the current audio stream transports encoded data, then he can put RXSTEO to 0, and DRFMT to 10 prior to start data reception. Note that DRFMT and RXSTEO cannot be modified when SPDIFRXEN = 11. Writes to these fields are ignored if SPDIFRXEN is already 11, though these field can be changed with the same write instruction that causes SPDIFRXEN to become 11.

Then the SPDIFRX waits for SPDIFRXEN = 11 and the “B” preamble before starting saving audio samples.
Figure 641. Synchronization flowchart

Initial Sync Process

Wait for 4 transitions if WFA = 1, else skip this step

Parallel flows

COARSE SYNC

Search for Longest/Shortest pulses for 70 transitions

Compute COARSE thresholds (THLO, THHI)

Search for preamble for 70 transitions

preamble found within 70 trans. ?

FINE SYNC

Symb. decoding (1) OK ?

Decode properly the next 40 symbols
Measurement of 24 and 40 symbols duration (WIDTH24, WIDTH40)

ATTEMPT ++

Compute FINE thresholds (THLO, THHI)

Set SYNCD to 1

ERROR: Sync failure !
TERR = 1

Sync stopped

ERROR: Sync failure !
SERR = 1

ATTEMPT == NBTR ?

ERROR: Sync failure !
TERR = 1

Sync stopped

ERROR: Sync failure !
SERR = 1

Y

TRCNT overflows ?

N

N

Y

Y

(1) - The decoding is considered OK, when the symbols are properly decoded, and preamble occurs at the expected position

Refer to Frame structure and synchronization error for additional information concerning TRCNT overflow.

The FINE SYNC process is re-triggered every frame in order to update thresholds as shown in Figure 642 in order to continuously track S/PDIF synchronization.
57.3.6 **SPDIFRX handling**

The software can control the state of the SPDIFRX through SPDIFRXEN field. The SPDIFRX can be into one of the following states:

- **STATE_IDLE**: The peripheral is disabled, the spdifrx_ker_ck domain is reset. The spdifrx_pclk domain is functional.
- **STATE_SYNC**: The peripheral is synchronized to the stream, thresholds are updated regularly, user and channel status can be read via interrupt of DMA. The audio samples are not provided to receive buffer.
- **STATE_RCV**: The peripheral is synchronized to the stream, thresholds are updated regularly, user, channel status and audio samples can be read via interrupt or DMA channels. When SPDIFRXEN goes to 11, the SPDIFRX waits for “B” preamble before starting saving audio samples.
- **STOP_STATE**: The peripheral is no longer synchronized, the reception of the user, channel status and audio samples are stopped. It is expected that the software re-starts the SPDIFRX.

The Figure 643 shows the possible states of the SPDIFRX, and how to transition from one state to the other. The bits under software control are followed by the mention “(SW)”, the bits under IP control are followed by the mention “(HW)”.

![Figure 642. Synchronization process scheduling](image)
When SPDIFRX is in STATE_IDLE:
- The software can transition to STATE_SYNC by setting SPDIFRXEN to 01 or 11

When SPDIFRX is in STATE_SYNC:
- If the synchronization fails or if the received data are not properly decoded with no chance of recovery without a re-synchronization (FERR or SERR or TERR = 1), the SPDIFRX goes to STATE_STOP, and waits for software acknowledge.
- When the synchronization phase is completed, if SPDIFRXEN = 01 the peripheral remains in this state.
- At any time the software can set SPDIFRXEN to 0, then SPDIFRX returns immediately to STATE_IDLE. If a DMA transfer is on-going, it will properly be completed.
- The SPDIFRX goes to STATE_RCV if SPDIFRXEN = 11 and if the SYNCD = 1

When SPDIFRX is in STATE_RCV:
- If the received data are not properly decoded with no chance of recovery without a re-synchronization (FERR or SERR or TERR = 1), the SPDIFRX goes to STATE_STOP, and waits for software acknowledge.
- At any time the software can set SPDIFRXEN to 0, then SPDIFRX returns immediately to STATE_IDLE. If a DMA transfer is on-going, it will properly be completed.

When SPDIFRX is in STATE_STOP:
- The SPDIFRX stops reception and synchronization, and waits for the software to set the bit SPDIFRXEN to 0, in order to clear the error flags.

**NOTE:** SYNCD is an internal event informing that the SPDIFRX is properly synchronized.
57.3.7 Data reception management

The SPDIFRX offers a double buffer for the audio sample reception. A 32-bit buffer located into the spdifrx_ker_ck clock domain (RX_BUF), and the SPDIFRX_FMTx_DR register. The valid data contained into the RX_BUF are immediately transferred into SPDIFRX_FMTx_DR if SPDIFRX_FMTx_DR is empty.

The valid data contained into the RX_BUF are transferred into SPDIFRX_FMTx_DR when the two following conditions are reached:

- The transition between the parity bit (P) and the next preamble is detected (this indicated that the word has been completely received).
- The SPDIFRX_FMTx_DR is empty.

Having a 2-word buffer gives more flexibility for the latency constraint.

The maximum latency allowed is $T_{\text{SAMPLE}} - 2T_{\text{PCLK}} - 2T_{\text{spdifrx_ker_ck}}$

Where $T_{\text{SAMPLE}}$ is the audio sampling rate of the received stereo audio samples, $T_{\text{PCLK}}$ is the period of spdifrx_pclk clock, and $T_{\text{spdifrx_ker_ck}}$ is the period of spdifrx_ker_ck clock.

The SPDIFRX offers the possibility to use either DMA (spdifrx_dat_dma and spdifrx_ctrl_dma) or interrupts for transferring the audio samples into the memory. The recommended option is DMA, refer to Section 57.3.12: DMA Interface for additional information.

The SPDIFRX offers several way on handling the received data. The user can either have a separate flow for control information and audio samples, or get them all together.

For each sub-frame, the data reception register SPDIFRX_FMTx_DR contains the 24 data bits, and optionally the V, U, C, PE status bits, and the PT (see Mixing data and control flow).

Note that PE bit stands for parity error bit, and is set to 1 when a parity error is detected in the decoded sub-frame.

The PT field carries the preamble type (B, M or W).

V, U and C are a direct copy of the value received from the S/PDIF interface.

The bit DRFMT allows the selection between 3 audio formats as shown in Figure 644.

This document describes 3 data registers: SPDIFRX_FMTx[2:0] ($x = 2$ to $0$), but in reality there is only one physical data register, having 3 possible formats:

- When DRFMT = 0, the format of the data register is the one described by SPDIFRX_FMT0_DR
- When DRFMT = 1, the format of the data register is the one described by SPDIFRX_FMT1_DR
- When DRFMT = 2, the format of the data register is the one described by SPDIFRX_FMT2_DR
Setting DRFMT to 00 or 01, offers the possibility to have the data either right or left aligned into the SPDIFRX_FMTx_DR register. The status information can be enabled or forced to zero according to the way the software wants to handle them.

The format given by DRFMT= 10 is interesting in non-linear mode, as only 16 bits per sub-frame are used. By using this format, the data of two consecutive sub-frames are stored into SPDIFRX_FMTx_DR, dividing by two the amount of memory footprint. Note that when RXSTEO = 1, there is no misalignment risks (i.e. data from ChA are always stored into SPDIFRX_FMTx_DR[31:16]). If RXSTEO = 0, then there is a misalignment risk is case of overrun situation. In that case SPDIFRX_FMTx_DR[31:16] always contain the oldest value and SPDIFRX_FMTx_DR[15:0] the more recent value (see Figure 646).

In this format the status information cannot be mixed with data, but the user can still get them through SPDIFRX_CSR register, and use a dedicated DMA channel or interrupt to transfer them to memory (see Section 57.3.8: Dedicated control flow).
Mixing data and control flow

The user can choose to use this mode in order to get the full flexibility of the handling of the control flow. The user can select which field shall be kept into the data register (SPDIFRX_FMTx_DR).

- When bit PMSK = 1, the Parity Error information is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.
- When bit VMSK = 1, the Validity information is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.
- When bit CUMSK = 1, the Channel Status, and Used data information are masked (set to 0), otherwise they are copied into SPDIFRX_FMTx_DR.
- When bit PTMSK = 1, the Preamble Type is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.

57.3.8 Dedicated control flow

The SPDIFRX offers the possibility to catch both user data and channel status information via a dedicated DMA channel. This feature allows the SPDIFRX to acquire continuously the channel status and user information. The acquisition starts at the beginning of a IEC 60958 block. Two fields are available to control this path: CBDMAEN and SPDIFRXEN. When SPDIFRXEN is set to 01 or 0x11, the acquisition is started, after completion of the synchronization phase. When 8 channel status and 16 user data bits have been received, they are packed and stored into SPDIFRX_CSR register. A DMA request is triggered if the bit CBDMAEN is set to 1 (see Figure 645).

If CS[0] corresponds to the first bit of a new block, the bit SOB is set to 1. Refer to Section 57.5.8: Channel status register (SPDIFRX_CSR). A bit is available (CHSEL) in order to select if the user wants to select channel status information (C) from the channel A or B.

Figure 645. Channel/user data format

Note: Once the first start of block is detected (B preamble), the SPDIFRX is checking the preamble type every 8 frames.

Note: Overrun error on SPDIFRX_FMTx_DR register does not affect this path.
57.3.9 Reception errors

Frame structure and synchronization error

The SPDIFRX, detects errors, when one of the following condition occurs:

- The FERR bit is set to 1 on the following conditions:
  - For each of the 28 information bits, if one symbol transition sequence is not correct: for example if short pulses are not grouped by pairs.
  - If preambles occur to an unexpected place, or an expected preamble is not received.
- The SERR bit is set when the synchronization fails, because the number of re-tries exceeded the programmed value.
- The TERR bit is set when the counter used to estimate the width between two transitions overflows (TRCNT).
  The overflow occurs when no transition is detected during 8192 periods of spdifrx_ker_ck clock. It represents at most a time interval of 11.6 frames.

When one of those flags goes to 1, the traffic on selected SPDIFRX_IN is then ignored, an interrupt is generated if the IFEIE bit of the SPDIFRX_CR register is set.

The normal procedure when one of those errors occur is:
- Set SPDIFRXEN to 0 in order to clear the error flags
- Set SPDIFRXEN to 01 or 11 in order to restart the IP

Refer to Figure 643 for additional information.

Parity error

For each sub-frame, an even number of zeros and ones is expected inside the 28 information bits. If not, the parity error bit PERR is set in the SPDIFRX_SR register and an interrupt is generated if the parity interrupt enable PERRIE bit is set in the SPDIFRX_CR register. The reception of the incoming data is not paused, and the SPDIFRX continue to deliver data to SPDIFRX_FMTx_DR even if the interrupt is still pending.

The interrupt is acknowledged by clearing the PERR flag through PERRCF bit.

If the software wants to guarantee the coherency between the data read in the SPDIFRX_FMTx_DR register and the value of the bit PERR, the bit PMSK must be set to 0.

Overrun error

If both SPDIFRX_FMTx_DR and RX_BUF are full, while the SPDIFRX_DC needs to write a new sample in RX_BUF, this new sample is dropped, and an overrun condition is triggered.

The overrun error flag OVR is set in the SPDIFRX_SR register and an interrupt is generated if the OVRIE bit of the SPDIFRX_CR register is set.

If the RXSTEO bit is set to 0, then as soon as the RX_BUF is empty, the IP stores the next incoming data, even if the OVR flag is still pending. The main purpose is to reduce as much as possible the amount of lost samples. Note that the behavior is similar independently of DRFMT value. See Figure 646.
If the RXSTEO bit is set to 1, it means that stereo data are transported, then the SPDIFRX has to avoid misalignment between left and right channels. So the peripheral has to drop a second sample even if there is room inside the RX_BUF in order to avoid misalignment. Then the incoming samples can be written normally into the RX_BUF even if the OVR flag is still pending. Refer to Figure 647.

The OVR flag is cleared by software, by setting the OVR CF bit to 1.
Figure 647. S/PDIF overrun error when RXSTEO = 1

57.3.10 Clocking strategy

The SPDIFRX block needs two different clocks:

- The APB clock (spdifrx_pclk), which is used for the register interface,
- The spdifrx_ker_ck which is mainly used by the SPDIFRX_DC part. Those clocks are not supposed to be phase locked, so all signals crossing those clock domains are re-synchronized (SYNC block on Figure 633).

In order to decode properly the incoming S/PDIF stream the SPDIFRX_DC shall re-sample the received data with a clock at least 11 times higher than the maximum symbol rate, or 704 times higher than the audio sample rate. For example if the user expects to receive a symbol rate to up to 12.288 MHz, the sample rate shall be at least 135.2 MHz. The clock used by the SPDIFRX_DC is the spdifrx_ker_ck.

The frequency of the spdifrx_pclk must be at least equal to the symbol rate.

Table 418. Minimum spdifrx_ker_ck frequency versus audio sampling rate

<table>
<thead>
<tr>
<th>Symbol Rate</th>
<th>Minimum spdifrx_ker_ck frequency</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.072 MHz</td>
<td>33.8 MHz</td>
<td>For 48 kHz stream</td>
</tr>
<tr>
<td>6.144 MHz</td>
<td>67.6 MHz</td>
<td>For 96 kHz stream</td>
</tr>
<tr>
<td>12.288 MHz</td>
<td>135.2 MHz</td>
<td>For 192 kHz stream</td>
</tr>
</tbody>
</table>
57.3.11 Symbol clock generation

The SPDIFRX block provides a symbol clock on signal named `spdifrx_symb_ck`, which can be used as the reference kernel clock for another audio device such as SAI or SPI/I2S. It could be used for SPDIFRX to I2S bridge function.

The symbol clock is built using the values of WIDTH24, WIDTH40 and the symbol boundaries.

- During the reception of the sub-frame sync preambles, the falling and rising edges of the symbol clock are built from the WIDTH24 and WIDTH40 values. Note that WIDTH24 and WIDTH40 are also used for the generation of the symbol clock, when the SPDIFRX is STATE_STOP or STATE_IDLE. See Table 419 for details.

- During the reception of the sub-frame payload, the SPDIFRX uses the symbols boundaries to generate the rising edge, the WIDTH24 and WIDTH40 values for the generation of the falling edge.

The duty cycle of the symbol clock is close to 50% during the reception of the sub-frame payload. However, the duty cycle can be altered when the SPDIFRX transitions from a symbol clock generated with WIDTH24 and WIDTH40 to a clock generated by the symbol clock boundaries or vice-versa.

The symbol clock will have an important jitter mainly due to:

- The re-sampling of the S/PDIF signal with `spdifrx_ker_ck` clock
- The transition of the symbol clock generation mode

For that reason the application shall consider the quality degradation if the symbol clock is used as the reference clock for A/D or D/A converters.

The generation of this symbol clock is controlled by the CKSEN bit. When CKSEN = ‘1’, the clock symbol is generated when the SPDIFRX completes successfully the first fine synchronization (SYNCD = 1), and when it is receiving correct data from the selected SPDIFRX input.

When the SPDIFRX goes to STATE_STOP, or STATE_IDLE, the symbol clock is gated if the bit CKSBKPEN = ‘0’. If the CKSBKPEN = ‘1’, then a backup symbol clock is still generated if the SPDIFRX is properly synchronized (i.e. valid values available for WIDTH24 and WIDTH40). Table 419 gives more details on the conditions controlling the generation of the symbol clock.
Note that when the flag SERR is set to ‘1’, neither the symbol clock nor the backup clock can be generated, since there is no synchronization.

Note that when both CKSEN and CKSBKPEN are set to ‘1’, the symbol clock will loose some transitions when the SPDIFRX switches from STATE_SYNC or STATE_RCV to STATE_STOP, or STATE_IDLE.

The bits CKSEN and CKSBKPEN are located into Control register (SPDIFRX_CR).

### 57.3.12 DMA Interface

The SPDIFRX interface is able to perform communication using the DMA.

*Note:* The user must refer to product specifications for availability of the DMA controller.

The SPDIFRX offers two independent DMA channels:
- A DMA channel dedicated to the data transfer
- A DMA channel dedicated to the channel status and user data transfer

The DMA mode for the data can be enabled for reception by setting the RXDMAEN bit in the SPDIFRX_CR register. In this case, as soon as the SPDIFRX_FMTx_DR is not empty, the...
SPDIFRX interface sends a transfer request to the DMA. The DMA reads the data received through the SPDIFRX_FMTx_DR register without CPU intervention.

For the use of DMA for the control data refer to Section 57.3.8: Dedicated control flow.

### 57.3.13 Interrupt Generation

An interrupt line is shared between:

- Reception events for data flow (RXNE)
- Reception event for control flow (CSRNE)
- Data corruption detection (PERR)
- Transfer flow interruption (OVR)
- Frame structure and synchronization errors (SERR, TERR and FERR)
- Start of new block interrupt (SBD)
- Synchronization done (SYNCD)

#### Figure 648. SPDIFRX interface interrupt mapping diagram

#### Clearing interrupt source

- RXNE is cleared when SPDIFRX_FMTx_DR register is read
- CSRNE is cleared when SPDIFRX_CSR register is read
- FERR is cleared when SPDIFRXEN is set to 0
- SERR is cleared when SPDIFRXEN is set to 0
- TERR is cleared when SPDIFRXEN is set to 0
- Others are cleared through SPDIFRX_IFCR register

**Note:** The SBD event can only occur when the SPDIFRX is synchronized to the input stream (SYNCD = 1).
The SBD flag behavior is not guaranteed when the sub-frame which contains the B preamble is lost due to an overrun.

### 57.3.14 Register protection

The SPDIFRX block embeds some hardware protection to avoid erroneous use of control registers. The table hereafter shows the bit field properties according to the SPDIFRX state.

#### Table 420. Bit field property versus SPDIFRX state

<table>
<thead>
<tr>
<th>Registers</th>
<th>Field</th>
<th>SPDIFRXEN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>00 (STATE_IDLE)</td>
</tr>
<tr>
<td>SPDIFRX_CR</td>
<td>INSEL</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>WFA</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>NBTR</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>CHSEL</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>CBDMAEN</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>PTMSK</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>CUMSK</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>VMSK</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>PMSK</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>DRFMT</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>RXSTEO</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>RXDMAEN</td>
<td>rw</td>
</tr>
<tr>
<td>SPDIFRX_IMR</td>
<td>All fields</td>
<td>rw</td>
</tr>
</tbody>
</table>

The table clearly shows that fields such as INSEL must be programmed when the IP is in STATE_IDLE. In the others IP states, the hardware prevents writing to this field.

**Note:** *Even if the hardware allows the writing of CBDMAEN and RXDMAEN “on-the-fly”, it is not recommended to enable the DMA when the IP is already receiving data.*

**Note:** *Note that each of the mask bits (PMSK, VMSK, …) can be changed “on-the-fly” at any IP state, but any change does not affect data which is already being held in SPDIFRX_FMTx_DR.*

### 57.4 Programming procedures

The following example illustrates a complete activation sequence of the SPDIFRX block. The data path and channel status and user information both use a dedicated DMA channel. The activation sequence is then split into the following steps:

- Wait for valid data on the selected SPDIFRX_IN input
- Synchronize to the S/PDIF stream
- Read the channel status and user information in order to setup the complete audio path
- Start data acquisition
A simple way to check if valid data are available into the SPDIFRX_IN line is to switch the SPDIFRX into the STATE SYNC, with bit WFA set to 1. The description hereafter focuses on detection. It is also possible to implement this function as follow:

- The software has to check from time to time (i.e. every 100 ms for example) if the SPDIFRX can find synchronization. This can be done by checking if the bit TERR is set. When it is set it indicates that no activity as been found.
- Connect the SPDIFRX_IN input to an external interrupt event block in order to detect transitions of SPDIFRX_IN line. When activity is detected, then SPDIFRXEN can be set to 01 or 11.

For those two implementations, the bit WFA is set to 0.

57.4.1 Initialization phase

- The initialization function looks like this:
- Configure the DMA transfer for both audio samples and IEC60958 channel status and user information (DMA channel selection and activation, priority, number of data to transfer, circular/no circular mode, DMA interrupts)
- Configure the destination address:
  - Configure the address of the SPDIFRX_CSR register as source address for IEC60958 channel status and user information
  - Configure the address of the SPDIFRX_FMTx_DR register as source address for audio samples
  - Enable the generation of the spdifrx_ker_ck. Refer to Table 418 in order to define the minimum clock frequency versus supported audio sampling rate. Note that the audio sampling rate of the received stream is not known in advance. This means that the user has to select a spdifrx_ker_ck frequency at least 704 times higher than the maximum audio sampling rate the application is supposed to handle: for example if the application is able to handle streams to up to 96 kHz, then F_spdifrx_ker_ck shall be at least 704 x 96 kHz = 67.6 MHz
- Enable interrupt for errors and event signaling (IFEIE = SYNCDIE = OVRIE, PERRIE = 1, others set to 0). Note that SYNCDIE can be set to 0.
- Configure the SPDIFRX_CR register:
  - INSEL shall select the wanted input
  - NBTR = 2, WFA = 1 (16 re-tries allowed, wait for activity before going to synchronization phase),
  - PTMSK = CUMSK = 1 (Preamble, C and U bits are not mixed with data)
  - VMSK = PMSK = 0 (Parity error and validity bit mixed with data)
  - CHSEL = 0 (channels status are read from sub-frame A)
  - DRFMT = 01 (data aligned to the left)
  - RXSTEO = 1 (expected stereo mode linear)
  - CBDMAEN = RXDMAEN = 1 (enable DMA channels)
  - SPDIFRXEN = 01 (switch SPDIFRX to STATE_SYNC)
- The CPU can enter in WFI mode

Then the CPU receives interrupts coming either from DMA or SPDIFRX.
57.4.2 Handling of interrupts coming from SPDIFRX

When an interrupt from the SPDIFRX is received, then the software has to check what is the source of the interrupt by reading the SPDIFRX_SR register.

- If SYNCD is set to 1, then it means that the synchronization has been properly completed. No action has to be performed in our case as the DMA is already programmed. The software just needs to wait for DMA interrupt in order to read channel status information.
  The SYNCD flag must be cleared by setting SYNCDCF bit of SPDIFRX_IFCR register to 1.

- If TERR or SERR or FERR are set to 1, the software has to set SPDIFRXEN to 0, and re-start from the initialization phase.
  - TERR indicates that a time-out occurs either during synchronization phase or after.
  - SERR indicates that the synchronization fails because the maximum allowed re-tries have been reached.
  - FERR indicates that the reading of information after synchronization fails (unexpected preamble, bad data decoding...).

- If PERR is set to 1, it means that a parity error has been detected, so one of the received audio sample or the channel status or user data bits are corrupted. The action taken here depends on the application: one action could be to drop the current channel status block as it is not reliable. There is no need to re-start from the initialization phase, as the synchronization is not lost.
  The PERR flag must be cleared by setting PERRCF bit of SPDIFRX_IFCR register to 1.

57.4.3 Handling of interrupts coming from DMA

If an interrupt is coming from the DMA channel used of the channel status (SPDIFRX_CSR):

If no error occurred (i.e. PERR), the CPU can start the decoding of channel information. For example bit 1 of the channel status informs the user if the current stream is linear or not. This information is very important in order to set-up the proper processing chain. In the same way, bits 24 to 27 of the channel status give the sampling frequency of the stream incoming stream.

Thanks to that information, the user can then configure the RXSTEO bit and DRFMT field prior to start the data reception. For example if the current stream is non linear PCM then RXSTEO is set to 0, and DRFMT is set to 10. Then the user can enable the data reception by setting SPDIFRXEN to 11.

The bit SOB, when set to 1 indicates the start of a new block. This information helps the software to identify the bit 0 of the channel status. Note that if the DMA generates an interrupt every time 24 values are transferred into the memory, then the first word always corresponds to the start of a new block.

If an interrupt is coming from the DMA channel used of the audio samples (SPDIFRX_FMTx_DR):

The process performed here depends of the data type (linear or non-linear), and on the data format selected.

For example in linear mode, if PE or V bit is set a special processing can be performed locally in order to avoid spurs on output. In non-linear mode those bits are not important as data frame have their own checksum.
57.5   SPDIFRX interface registers

57.5.1  Control register (SPDIFRX_CR)

Only 32-bit accesses are allowed in this register.

Address offset: 0x00

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>CKSBKPEN</th>
<th>CKSEN</th>
<th>INSEL[2:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 21  **CKSBKPEN**: Backup Symbol Clock Enable

This bit is set/reset by software

1: The SPDIFRX generates a backup symbol clock if CKSEN = ‘1’

0: The SPDIFRX does not generate a backup symbol clock

Bit 20  **CKSEN**: Symbol Clock Enable

This bit is set/reset by software

1: The SPDIFRX generates a symbol clock

0: The SPDIFRX does not generate a symbol clock

Bit 19  Reserved, must be kept at reset value.

Bits 18:16  **INSEL[2:0]**: SPDIFRX input selection\(^{(1)}\)

000: SPDIFRX_IN1 selected

001: SPDIFRX_IN2 selected

010: SPDIFRX_IN3 selected

011: SPDIFRX_IN4 selected

others reserved

Bit 15  Reserved, must be kept at reset value.

Bit 14  **WFA**: Wait for activity\(^{(1)}\)

This bit is set/reset by software

1: The SPDIFRX waits for activity on SPDIFRX_IN line (4 transitions) before performing the synchronization

0: The SPDIFRX does not wait for activity on SPDIFRX_IN line before performing the synchronization
Bits 13:12 **NBTR[1:0]**: Maximum allowed re-tries during synchronization phase\(^{(1)}\)
- 00: No re-try is allowed (only one attempt)
- 01: 3 re-tries allowed
- 10: 15 re-tries allowed
- 11: 63 re-tries allowed

Bit 11 **CHSEL**: Channel selection\(^{(1)}\)
- This bit is set/reset by software
- 1: The control flow takes the channel status from channel B
- 0: The control flow takes the channel status from channel A

Bit 10 **CBDMAEN**: Control buffer DMA enable for control flow\(^{(1)}\)
- This bit is set/reset by software
- 1: DMA mode is enabled for reception of channel status and used data information.
- 0: DMA mode is disabled for reception of channel status and used data information.

When this bit is set, the DMA request is made whenever the CSRNE flag is set.

Bit 9 **PTMSK**: Mask of preamble type bits\(^{(1)}\)
- This bit is set/reset by software
- 1: The preamble type bits are not copied into the SPDIFRX_FMTx_DR, zeros are written instead
- 0: The preamble type bits are copied into the SPDIFRX_FMTx_DR

Bit 8 **CUMSK**: Mask of channel status and user bits\(^{(1)}\)
- This bit is set/reset by software
- 1: The channel status and user bits are not copied into the SPDIFRX_FMTx_DR, zeros are written instead
- 0: The channel status and user bits are copied into the SPDIFRX_FMTx_DR

Bit 7 **VMSK**: Mask of validity bit\(^{(1)}\)
- This bit is set/reset by software
- 1: The validity bit is not copied into the SPDIFRX_FMTx_DR, a zero is written instead
- 0: The validity bit is copied into the SPDIFRX_FMTx_DR

Bit 6 **PMSK**: Mask parity error bit\(^{(1)}\)
- This bit is set/reset by software
- 1: The parity error bit is not copied into the SPDIFRX_FMTx_DR, a zero is written instead
- 0: The parity error bit is copied into the SPDIFRX_FMTx_DR

Bits 5:4 **DRFMT[1:0]**: RX data format\(^{(1)}\)
- This bit is set/reset by software
- 11: reserved
- 10: Data sample are packed by setting two 16-bit sample into a 32-bit word
- 01: Data samples are aligned in the left (MSB)
- 00: Data samples are aligned in the right (LSB)
Bit 3  **RXSTEO**: Stereo mode\(^{(1)}\)
    This bit is set/reset by software
    1: The peripheral is in STEREO mode
    0: The peripheral is in MONO mode
    This bit is used in case of overrun situation in order to handle misalignment

Bit 2  **RXDMAEN**: Receiver DMA enable for data flow\(^{(1)}\)
    This bit is set/reset by software
    1: DMA mode is enabled for reception.
    0: DMA mode is disabled for reception.
    When this bit is set, the DMA request is made whenever the RXNE flag is set.

Bits 1:0  **SPDIFRXEN[1:0]**: Peripheral block enable\(^{(1)}\)
    This field is modified by software.
    It shall be used to change the peripheral phase among the three possible states: STATE_IDLE,
    STATE.Sync and STATE.RCV.

    00: Disable SPDIFRX (STATE_IDLE).
    01: Enable SPDIFRX Synchronization only
    10: Reserved
    11: Enable SPDIF Receiver

**Note**: it is not possible to transition from STATE_RCV to STATE_SYNC, the user shall first go the
STATE_IDLE.

    it is possible to transition from STATE_IDLE to STATE_RCV: in that case the peripheral
    transitions from STATE_IDLE to STATE_SYNC and as soon as the synchronization is
    performed goes to STATE_RCV.

1. Refer to Section 57.3.14: Register protection for additional information on fields properties.
57.5.2 Interrupt mask register (SPDIFRX_IMR)

Address offset: 0x04
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>Bit 31:7</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>IFEIE: Serial Interface Error Interrupt Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Interrupt is inhibited</td>
<td></td>
</tr>
<tr>
<td>1: A SPDIFRX interface interrupt is generated whenever SERR=1, TERR=1 or FERR=1 in the SPDIFRX_SR register.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 5</th>
<th>SYNCDE: Synchronization Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Interrupt is inhibited</td>
<td></td>
</tr>
<tr>
<td>1: A SPDIFRX interface interrupt is generated whenever SYNC = 1 in the SPDIFRX_SR register.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>SBLKIE: Synchronization Block Detected Interrupt Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Interrupt is inhibited</td>
<td></td>
</tr>
<tr>
<td>1: A SPDIFRX interface interrupt is generated whenever SBD = 1 in the SPDIFRX_SR register.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>OVRIE: Overrun error Interrupt Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Interrupt is inhibited</td>
<td></td>
</tr>
<tr>
<td>1: A SPDIFRX interface interrupt is generated whenever OVR=1 in the SPDIFRX_SR register.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>PERRIE: Parity error interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Interrupt is inhibited</td>
<td></td>
</tr>
<tr>
<td>1: A SPDIFRX interface interrupt is generated whenever PERR=1 in the SPDIFRX_SR register.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>CSRNEIE: Control Buffer Ready Interrupt Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Interrupt is inhibited</td>
<td></td>
</tr>
<tr>
<td>1: A SPDIFRX interface interrupt is generated whenever CSRNE = 1 in the SPDIFRX_SR register.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>RXNEIE: RXNE interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Interrupt is inhibited</td>
<td></td>
</tr>
<tr>
<td>1: A SPDIFRX interface interrupt is generated whenever RXNE=1 in the SPDIFRX_SR register.</td>
<td></td>
</tr>
</tbody>
</table>
57.5.3 Status register (SPDIFRX_SR)

Address offset: 0x08
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>must be kept at reset value.</td>
</tr>
</tbody>
</table>
| 30:16 | WIDTH5[14:0] | duration of 5 symbols counted with spdifrx_ker_ck. This value represents the amount of spdifrx_ker_ck clock periods contained on a length of 5 consecutive symbols. It can be used to estimate the S/PDIF symbol rate. Its accuracy is limited by the frequency of spdifrx_ker_ck. For example, if the spdifrx_ker_ck is fixed to 84 MHz, and WIDTH5 = 147d. The estimated sampling rate of the S/PDIF stream is: 
Fs = 5 x Fspdifrx_ker_ck / (WIDTH5 x 64) ~ 44.6 kHz, so the closest standard sampling rate is 44.1 kHz. Note that WIDTH5 is updated by the hardware when SYNCD goes high, and then every frame. |
| 15:9 | Reserved | must be kept at reset value. |
| 8  | TERR | time-out error. This bit is set by hardware when the counter TRCNT reaches its max value. It indicates that the time interval between two transitions is too long. It generally indicates that there is no valid signal on SPDIFRX_IN input. This flag is cleared by writing SPDIFRXEN to 0. An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register. 0: No sequence error is detected 1: Sequence error is detected |
| 7  | SERR | synchronization error. This bit is set by hardware when the synchronization fails due to amount of re-tries for NBTR. This flag is cleared by writing SPDIFRXEN to 0. An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register. 0: No synchronization error is detected 1: Synchronization error is detected |
| 6  | FERR | framing error. This bit is set by hardware when an error occurs during data reception: preamble not at the expected place, short transition not grouped by pairs... This bit is set by the hardware only if the synchronization has been completed (SYNCD = 1). This flag is cleared by writing SPDIFRXEN to 0. An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register. 0: no Manchester Violation detected 1: Manchester Violation detected |
Bit 5 **SYNCD**: synchronization done
   This bit is set by hardware when the initial synchronization phase is properly completed.
   This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.
   An interrupt is generated if SYNC DIE = 1 in the SPDIFRX_IMR register.
   0: Synchronization is pending
   1: Synchronization is completed

Bit 4 **SBD**: synchronization block detected
   This bit is set by hardware when a “B” preamble is detected
   This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.
   An interrupt is generated if SBLKIE = 1 in the SPDIFRX_IMR register.
   0: No “B” preamble detected
   1: “B” preamble has been detected

Bit 3 **OVR**: overrun error
   This bit is set by hardware when a received data is ready to be transferred in the SPDIFRX_FMTx_DR register while RXNE = 1 and both SPDIFRX_FMTx_DR and RX_BUF are full.
   This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.
   An interrupt is generated if OVRIE=1 in the SPDIFRX_IMR register.
   0: No Overrun error
   1: Overrun error is detected

*Note: When this bit is set, the SPDIFRX_FMTx_DR register content is not lost but the last data received are.*

Bit 2 **PERR**: parity error
   This bit is set by hardware when the data and status bits of the sub-frame received contain an odd number of 0 and 1.
   This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.
   An interrupt is generated if PIE = 1 in the SPDIFRX_IMR register.
   0: No parity error
   1: Parity error

Bit 1 **CSRNE**: the control buffer register is not empty
   This bit is set by hardware when a valid control information is ready.
   This flag is cleared when reading SPDIFRX_CSR register.
   An interrupt is generated if CBRDYIE = 1 in the SPDIFRX_IMR register
   0: No control word available on SPDIFRX_CSR register
   1: A control word is available on SPDIFRX_CSR register

Bit 0 **RXNE**: read data register not empty
   This bit is set by hardware when a valid data is available into SPDIFRX_FMTx_DR register.
   This flag is cleared by reading the SPDIFRX_FMTx_DR register.
   An interrupt is generated if RXNEIE=1 in the SPDIFRX_IMR register.
   0: Data is not received
   1: Received data is ready to be read.
57.5.4 Interrupt flag clear register (SPDIFRX_IFCR)

Address offset: 0x0C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:6 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 5 <strong>SYNCDCF</strong>: clears the synchronization done flag</td>
</tr>
<tr>
<td>Writing 1 in this bit clears the flag SYNCD in the SPDIFRX_SR register.</td>
</tr>
<tr>
<td>Reading this bit always returns the value 0.</td>
</tr>
<tr>
<td>Bit 4 <strong>SBDCF</strong>: clears the synchronization block detected flag</td>
</tr>
<tr>
<td>Writing 1 in this bit clears the flag SBD in the SPDIFRX_SR register.</td>
</tr>
<tr>
<td>Reading this bit always returns the value 0.</td>
</tr>
<tr>
<td>Bit 3 <strong>OVRCF</strong>: clears the overrun error flag</td>
</tr>
<tr>
<td>Writing 1 in this bit clears the flag OVR in the SPDIFRX_SR register.</td>
</tr>
<tr>
<td>Reading this bit always returns the value 0.</td>
</tr>
<tr>
<td>Bit 2 <strong>PERRCF</strong>: clears the parity error flag</td>
</tr>
<tr>
<td>Writing 1 in this bit clears the flag PERR in the SPDIFRX_SR register.</td>
</tr>
<tr>
<td>Reading this bit always returns the value 0.</td>
</tr>
<tr>
<td>Bits 1:0 Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
57.5.5 Data input register (SPDIFRX_FMT0_DR)

Address offset: 0x10
Reset value: 0x0000 0000

This register can take 3 different formats according to DRFMT. Here is the format when DRFMT = 00:

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

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

Bits 29:28  **PT[1:0]:** preamble type

These bits indicate the preamble received.

- 00: not used
- 01: Preamble B received
- 10: Preamble M received
- 11: Preamble W received

Note that if PTMSK = 1, this field is forced to zero

Bit 27  **C:** channel status bit
Contains the received channel status bit, if CUMSK = 0, otherwise it is forced to 0

Bit 26  **U:** user bit
Contains the received user bit, if CUMSK = 0, otherwise it is forced to 0

Bit 25  **V:** validity bit
Contains the received validity bit if VMSK = 0, otherwise it is forced to 0

Bit 24  **PE:** parity error bit
Contains a copy of PERR bit if PMSK = 0, otherwise it is forced to 0

Bits 23:0  **DR[23:0]:** data value
Contains the 24 received data bits, aligned on D[23]
57.5.6  Data input register (SPDIFRX_FMT1_DR)

Address offset: 0x10  
Reset value: 0x0000 0000

This register can take 3 different formats according to DRFMT. Here is the format when DRFMT = 01:

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

Bits 31:8  **DR[23:0]**: data value  
Contains the 24 received data bits, aligned on D[23]

Bits 7:6  Reserved, must be kept at reset value.

Bits 5:4  **PT[1:0]**: preamble type  
These bits indicate the preamble received.

00: not used  
01: preamble B received  
10: preamble M received  
11: preamble W received  

Note that if PTMSK = 1, this field is forced to zero

Bit 3  **C**: channel Status bit  
Contains the received channel status bit, if CUMSK = 0, otherwise it is forced to 0

Bit 2  **U**: user bit  
Contains the received user bit, if CUMSK = 0, otherwise it is forced to 0

Bit 1  **V**: validity bit  
Contains the received validity bit if VMSK = 0, otherwise it is forced to 0

Bit 0  **PE**: parity error bit  
Contains a copy of PERR bit if PMSK = 0, otherwise it is forced to 0
57.5.7 Data input register (SPDIFRX_FMT2_DR)

Address offset: 0x10
Reset value: 0x0000 0000

This register can take 3 different formats according to DRFMT.

The data format proposed when DRFMT = 10, is dedicated to non-linear mode, as only 16 bits are used (bits 23 to 8 from S/PDIF sub-frame).

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

Bits 31:16 DRNL2[15:0]: data value
- This field contains the channel A

Bits 15:0 DRNL1[15:0]: data value
- This field contains the channel B
57.5.8 Channel status register (SPDIFRX_CSR)

Address offset: 0x14
Reset value: 0x0000 0000

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SOB**: start of block

This bit indicates if the bit CS[0] corresponds to the first bit of a new block
0: CS[0] is not the first bit of a new block
1: CS[0] is the first bit of a new block

Bits 23:16 **CS[7:0]**: channel A status information

Bit CS[0] is the oldest value

Bits 15:0 **USR[15:0]**: user data information

Bit USR[0] is the oldest value, and comes from channel A, USR[1] comes channel B.
So USR[n] bits come from channel A is n is even, otherwise they come from channel B.

57.5.9 Debug information register (SPDIFRX_DIR)

Address offset: 0x18
Reset value: 0x0000 0000

Bits 31:29 Reserved, must be kept at reset value.
Bits 28:16 **TLO[12:0]**: threshold LOW (TLO = 1.5 x UI / T_{spdifrx_ker_ck})

This field contains the current threshold LOW estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of TLO is limited to a period of the spdifrx_ker_ck. The sampling rate can be estimated as follow:

Sampling Rate = \[2 \times TLO \times T_{spdifrx_ker_ck} \pm T_{spdifrx_ker_ck}\] x 2/3

Note that TLO is updated by the hardware when SYNCD goes high, and then every frame.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **THI[12:0]**: threshold HIGH (THI = 2.5 x UI / T_{spdifrx_ker_ck})

This field contains the current threshold HIGH estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of THI is limited to a period of the spdifrx_ker_ck. The sampling rate can be estimated as follow:

Sampling Rate = \[2 \times THI \times T_{spdifrx_ker_ck} \pm T_{spdifrx_ker_ck}\] x 2/5

Note that THI is updated by the hardware when SYNCD goes high, and then every frame.
### 57.5.10 SPDIFRX interface register map

*Table 421* gives the SPDIFRX interface register map and reset values.

**Table 421. SPDIFRX interface register map and reset values**

| Offset | Register name   | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | SPDIFRX_CR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x04   | SPDIFRX_IMR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x08   | SPDIFRX_SR     | WIDTH[14:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0C   | SPDIFRX_IFCR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x10   | SPDIFRX_FMT0_DR| PT[1:0] | C  | U  | V  | PE | DR[23:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x10   | SPDIFRX_FMT1_DR|     |    |    |    |    | DR[23:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x10   | SPDIFRX_FMT2_DR|     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x14   | SPDIFRX_CSR    | SB  | CS[7:0] |    |    | USR[15:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x18   | SPDIFRX_DIR    | TLO[12:0] |    |    |    |    | TH[12:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to *Section 2.3 on page 128* for the register boundary addresses.
58 Single Wire Protocol Master Interface (SWPMI)

58.1 Introduction

The Single Wire Protocol Master Interface (SWPMI) is the master interface corresponding to the Contactless front-end (CLF) defined in the ETSI TS 102 613 technical specification.

The principle of the Single wire protocol (SWP) is based on the transmission of digital information in full duplex mode:

- S1 signal (from Master to Slave) is transmitted by a digital modulation (L or H) in the voltage domain (refer to Figure 649: S1 signal coding).
- S2 signal (from Slave to Master) is transmitted by a digital modulation (L or H) in the current domain (refer to Figure 650: S2 signal coding).
58.2 SWPMI main features

The SWPMI module main features are the following (see Figure 58.3.4: SWP bus states):

- Full-duplex communication mode
- Automatic SWP bus state management
- Automatic handling of Start of frame (SOF)
- Automatic handling of End of frame (EOF)
- Automatic handling of stuffing bits
- Automatic CRC-16 calculation and generation in transmission
- Automatic CRC-16 calculation and checking in reception
- 32-bit Transmit data register
- 32-bit Receive data register
- Multi software buffer mode for efficient DMA implementation and multi frame buffering
- Configurable bit-rate up to 2 Mbit/s
- Configurable interrupts
- CRC error, underrun, overrun flags
- Frame reception and transmission complete flags
- Slave resume detection flag
- Loopback mode for test purpose
- Embedded SWPMI_IO transceiver compliant with ETSI TS 102 613 technical specification
- Dedicated mode to output SWPMI_RX, SWPMI_TX and SWPMI_SUSPEND signals on GPIOs, in case of external transceiver connection
58.3 SWPMI functional description

58.3.1 SWPMI block diagram

Refer to the bit SWPSEL in Section 8.7.18: RCC CPU domain kernel clock configuration register (RCC_CDCCIP1R) to select the swpmi_ker_ck (SWPMI core clock source).

Note: In order to support the exit from Stop mode by a RESUME by slave, it is mandatory to select HSI for swpmi_ker_ck. If this feature is not required, swpmi_pclk can be selected, and SWPMI must be disabled before entering the Stop mode.

58.3.2 SWPMI pins and internal signals

Table 422 lists the SWPMI slave inputs and output signals connected to package pins or balls, while Table 423 shows the internal SWPMI signals.

Table 422. SWPMI input/output signals connected to package pins or balls

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWPMI_SUSPEND</td>
<td>Digital output</td>
<td>SWPMI suspend signal</td>
</tr>
<tr>
<td>SWPMI_TX</td>
<td>Digital output</td>
<td>SWPMI transmit signal</td>
</tr>
</tbody>
</table>
58.3.3 SWP initialization and activation

The initialization and activation will set the SWPMI_IO state from low to high.

When using the internal transceiver, the procedure is the following:

1. Configure the SWP_CLASS bit in SWPMI_OR register according to the VDD voltage (3 V or 1.8 V),
2. Set SWPTEN in SWPMI_CR register to enable the SWPMI_IO transceiver and set the SWPMI_IO to low level (SWP bus DEACTIVATED)
3. Wait for the RDYF flag in SWPMI_SR register to be set (polling the flag or enabling the interrupt with RDYIE bit in SWPMI_IER register),
4. Set SWPACT bit in SWPMI_CR register to ACTIVATE the SWP i.e. to move from DEACTIVATED to SUSPENDED.

### Table 422. SWPMI input/output signals connected to package pins or balls

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWPMI_RX</td>
<td>Digital input</td>
<td>SWPMI receive signal</td>
</tr>
<tr>
<td>SWPMI_IO</td>
<td>Input and output</td>
<td>Internal SWPMI transceiver.</td>
</tr>
</tbody>
</table>

### Table 423. SWPMI internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>swpmi_pclk</td>
<td>Digital input</td>
<td>APB clock</td>
</tr>
<tr>
<td>swpmi_ker_ck</td>
<td>Digital input</td>
<td>SWPMI kernel clock</td>
</tr>
<tr>
<td>swpmi_wkup</td>
<td>Digital output</td>
<td>SWPMI wakeup signal</td>
</tr>
<tr>
<td>swpmi_gbl_it</td>
<td>Digital output</td>
<td>SWPMI interrupt signal</td>
</tr>
<tr>
<td>swpmi_tx_dma</td>
<td>Digital output</td>
<td>SWPMI DMA transmit request</td>
</tr>
<tr>
<td>swpmi_rx_dma</td>
<td>Digital output</td>
<td>SWPMI DMA receive request</td>
</tr>
</tbody>
</table>
58.3.4 SWP bus states

The SWP bus can have the following states: DEACTIVATED, SUSPENDED, ACTIVATED. Several transitions are possible:

- ACTIVATE: transition from DEACTIVATED to SUSPENDED state,
- SUSPEND: transition from ACTIVATED to SUSPENDED state,
- RESUME by master: transition from SUSPENDED to ACTIVATED state initiated by the master,
- RESUME by slave: transition from SUSPENDED to ACTIVATED state initiated by the slave,
- DEACTIVATE: transition from SUSPENDED to DEACTIVATED state.

ACTIVATE

During and just after reset, the SWPMI_IO is configured in analog mode. Refer to Section 58.3.3: SWP initialization and activation to activate the SWP bus.

SUSPEND

The SWP bus stays in the ACTIVATED state as long as there is a communication with the slave, either in transmission or in reception. The SWP bus switches back to the SUSPENDED state as soon as there is no more transmission or reception activity, after 7 idle bits.

RESUME by master

Once the SWPMI is enabled, the user can request a SWPMI frame transmission. The SWPMI first sends a transition sequence and 8 idle bits (RESUME by master) before starting the frame transmission. The SWP moves from the SUSPENDED to ACTIVATED state after the RESUME by master (refer to Figure 652: SWP bus states).

RESUME by slave

Once the SWPMI is enabled, the SWP can also move from the SUSPENDED to ACTIVATED state if the SWPMI receives a RESUME from the slave. The RESUME by slave sets the SRF flag in the SWPMI_ISR register.

DEACTIVATE

Deactivate request

If no more communication is required, and if SWP is in the SUSPENDED mode, the user can request to switch the SWP to the DEACTIVATED mode by disabling the SWPMI peripheral. The software must set DEACT bit in the SWPMI_CR register in order to request the DEACTIVATED mode. If no RESUME by slave is detected by SWPMI, the DEACTF flag is set in the SWPMI_ISR register and the SWPACT bit is cleared in the SWPMI_ICR register. In case a RESUME by slave is detected by the SWPMI while the software is setting DEACT bit, the SRF flag is set in the SWPMI_ISR register, DEACTF is kept cleared, SWPACT is kept set and DEACT bit is cleared.

In order to activate SWP again, the software must clear DEACT bit in the SWPMI_CR register before setting SWPACT bit.
Deactivate mode

In order to switch the SWP to the DEACTIVATED mode immediately, ignoring any possible incoming RESUME by slave, the user must clear SWPACT bit in the SWPMI_CR register.

Note: In order to further reduce current consumption, configure the SWPMI_IO port as output push pull low in GPIO controller and then clear the SWPTEN bit in SWPMI_CR register (refer to Section 11: General-purpose I/Os (GPIO)).

Figure 652. SWP bus states

58.3.5 SWPMI_IO (internal transceiver) bypass

A SWPMI_IO (transceiver), compliant with ETSI TS 102 613 technical specification, is embedded in the microcontroller. Nevertheless, this is possible to bypass it by setting SWP_TBYP bit in SWPMI_OR register. In this case, the SWPMI_IO is disabled and the SWPMI_RX, SWPMI_TX and SWPMI_SUSPEND signals are available as alternate functions on three GPIOs (refer to “Pinouts and pin description” in product datasheet). This configuration is selected to connect an external transceiver.

Note: In SWPMI_IO bypass mode, SWPTEN bit in SWPMI_CR register must be kept cleared.

58.3.6 SWPMI Bit rate

The bit rate must be set in the SWPMI_BRR register, according to the following formula:

$$F_{SWP} = F_{swpmi\_ker\_ck} / ((BR[7:0]+1)\times4)$$

Note: The maximum bitrate is 2 Mbit/s.
58.3.7 SWPMI frame handling

The SWP frame is composed of a Start of frame (SOF), a Payload from 1 to 30 bytes, a 16-bit CRC and an End of frame (EOF) (Refer to Figure 653: SWP frame structure).

![Figure 653. SWP frame structure](image)

The SWPMI embeds one 32-bit data register for transmission (SWPMI_TDR), and one 32-bit data register for reception (SWPMI_RDR).

In transmission, the SOF insertion, the CRC calculation and insertion, and the EOF insertion are managed automatically by the SWPMI. The user only has to provide the Payload content and size. A frame transmission starts as soon as data is written into the SWPMI_TDR register. Dedicated flags indicate an empty transmit data register and a complete frame transmission event.

In reception, the SOF deletion, the CRC calculation and checking, and the EOF deletion are managed automatically by the SWPMI. The user only has to read the Payload content and size. Dedicated flags indicate a full receive data register, a complete frame reception and possibly CRC error events.

The stuffing bits insertion (in transmission) and stuffing bits deletion (in reception) are managed automatically by the SWPMI core. These operations are transparent for the user.

58.3.8 Transmission procedure

Before starting any frame transmission, the user must activate the SWP. Refer to Section 58.3.3: SWP initialization and activation.

There are several possible software implementations for a frame transmission: No software buffer mode, Single software buffer mode, and Multi software buffer mode.

The software buffer usage requires the use of a DMA channel to transfer data from the software buffer in the RAM memory to the transmit data register in the SWPMI peripheral.

**No software buffer mode**

This mode does not require the use of DMA. The SWP frame transmission handling is done by polling status flags in the main loop or inside the SWPMI interrupt routine. There is a 32-bit transmit data register (SWPMI_TDR) in the SWPMI, thus writing to this register will trigger the transmission of up to 4 bytes.

The No software buffer mode is selected by clearing TXDMA bit in the SWPMI_CR register.

The frame transmission is started by the first write to the SWPMI_TDR register. The low significant byte of the first 32-bit word (bits [7:0]) written into the SWPMI_TDR register) indicates the number of data bytes in the payload, and the 3 other bytes of this word must
contain the first 3 bytes of the payload (bits [15:8] contain the first byte of the payload, bits [23:16] the second byte and bits [31:24] the third byte). Then, the following writes to the SWPMI_TDR register will only contain the following payload data bytes, up to 4 for each write.

**Note:** The low significant byte of the first 32-bit word written into the SWPMI_TDR register is coding the number of data bytes in the payload. This number could be from 1 to 30. Any other value in the low significant byte will be ignored and the transmission will not start.

Writing to the SWPMI_TDR register will induce the following actions:

- Send the transition sequence and 8 idle bits (RESUME by master) if the SWP bus state is SUPENDED (this will not happen if the SWP bus state is already ACTIVATED),
- Send a Start of frame (SOF),
- Send the payload according to the SWPMI_TRD register content. If the number of bytes in the payload is greater than 3, the SWPMI_TDR needs to be refilled by software, each time the TXE flag in the SWPMI_ISR register is set, and as long as the TXBEF flag is not set in the SWPMI_ISR register,
- Send the 16-bit CRC, automatically calculated by the SWPMI core,
- Send an End of frame (EOF).

The TXE flag is cleared automatically when the software is writing to the SWPMI_TDR register.

Once the complete frame is sent, provided that no other frame transmission has been requested (i.e. SWPMI_TDR has not been written again after the TXBEF flag setting), TCF and SUSP flags are set in the SWPMI_ISR register 7 idle bits after the EOF transmission, and an interrupt is generated if TCIE bit is set in the SWPMI_IER register (refer to [Figure 654: SWPMI No software buffer mode transmission](#)).

**Figure 654. SWPMI No software buffer mode transmission**

![Figure 654. SWPMI No software buffer mode transmission](image)

If another frame transmission is requested before the end of the EOF transmission, the TCF flag is not set and the frame will be consecutive to the previous one, with only one idle bit in between (refer to [Figure 655: SWPMI No software buffer mode transmission, consecutive frames](#)).
Single software buffer mode

This mode allows to transmit a complete SWP frame without a CPU intervention, using the DMA. The DMA will refill the 32-bit SWPMI_TDR register, and the software can poll the end of the frame transmission using the SWPMI_TXBEF flag.

The Single software buffer mode is selected by setting TXDMA bit and clearing TXMODE bit in the SWPMI_CR register.

The DMA channel or stream must be configured in following mode (refer to DMA section):
- memory to memory mode disabled,
- memory increment mode enabled,
- memory size set to 32-bit,
- peripheral size set to 32-bit,
- peripheral increment mode disabled,
- circular mode disabled,
- data transfer direction set to read from memory.
- the number of words to be transfered must be set according to the SWP frame length,
- the source address is the SWP frame buffer in RAM,
- the destination address is the SWPMI_TDR register.

Then the user must:
1. Set TXDMA bit in the SWPMI_CR register,
2. Set TXBEIE bit in the SWPMI_IER register,
3. Fill the buffer in the RAM memory (with the number of data bytes in the payload on the least significant byte of the first word),
4. Enable stream or channel in DMA module to start DMA transfer and frame transmission.

A DMA request is issued by SWPMI when TXE flag in SWPMI_ISR is set. The TXE flag is cleared automatically when the DMA is writing to the SWPMI_TDR register.
In the SWPMI interrupt routine, the user must check TXBEF bit in the SWPMI_ISR register. If it is set, and if another frame needs to be transmitted, the user must:
1. Disable stream or channel in DMA module
2. Update the buffer in the RAM memory with the content of the next frame to be sent
3. Configure the total number of words to be transferred in DMA module
4. Enable stream or channel in DMA module to start next frame transmission
5. Set CTXBEF bit in the SWPMI_ICR register to clear the TXBEF flag

**Multi software buffer mode**

This mode allows to work with several frame buffers in the RAM memory, in order to ensure a continuous transmission, keeping a very low CPU load, and allowing more latency for buffer update by software thanks to the DMA. The software can check the DMA counters at any time and update SWP frames accordingly in the RAM memory.

The Multi software buffer mode must be used in combination with DMA in circular mode. Each transmission buffer in the RAM memory must have a fixed length of eight 32-bit words, whatever the number of bytes in the SWP frame payload. The transmission buffers in the RAM memory must be filled by the software, keeping an offset of 8 between two consecutive ones. The first data byte of the buffer is the number of bytes of the frame payload. See the buffer example in *Figure 656: SWPMI Multi software buffer mode transmission*

The Multi software buffer mode is selected by setting both TXDMA and TXMODE bits in SWPMI_CR register.

For example, in order to work with 4 transmission buffers, the user must configure the DMA as follows:

The DMA channel or stream must be configured in following mode (refer to DMA section):
- memory to memory mode disabled,
- memory increment mode enabled,
- memory size set to 32-bit,
- peripheral size set to 32-bit,
- peripheral increment mode disabled,
- circular mode enabled,
- data transfer direction set to read from memory,
- the number of words to be transfered must be set to 32 (8 words per buffer),
- the source address is the buffer1 in RAM,
- the destination address is the SWPMI_TDR register.

Then, the user must:
1. Set TXDMA in the SWPMI_CR register
2. Set TXBEIE in the SWPMI_IER register
3. Fill buffer1, buffer2, buffer3 and buffer4 in the RAM memory (with the number of data bytes in the payload on the least significant byte of the first word)
4. Enable stream or channel in DMA module to start DMA.
In the SWPMI interrupt routine, the user must check TXBEF bit in the SWPMI_ISR register. If it is set, the user must set CTXBEF bit in SWPMI_ICR register to clear TXBEF flag and the user can update buffer1 in the RAM memory.

In the next SWPMI interrupt routine occurrence, the user will update buffer2, and so on.

The Software can also read the DMA counter (number of data to transfer) in the DMA registers in order to retrieve the frame which has already been transferred from the RAM memory and transmitted. For example, if the software works with 4 transmission buffers, and if the DMA counter equals 17, it means that two buffers are ready for updating in the RAM area. This is useful in case several frames are sent before the software can handle the SWPMI interrupt. If this happens, the software will have to update several buffers.

When there are no more frames to transmit, the user must disable the circular mode in the DMA module. The transmission will stop at the end of the buffer4 transmission.

If the transmission needs to stop before (for example at the end of buffer2), the user must set the low significant byte of the first word to 0 in buffer3 and buffer4.

TXDMA bit in the SWPMI_CR register will be cleared by hardware as soon as the number of data bytes in the payload is read as 0 in the least significant byte of the first word.

**Figure 656. SWPMI Multi software buffer mode transmission**

```plaintext
dma2_cmar2
```

Frame 1: TFL=8

Frame 2: TFL=18

Frame 3: TFL=30

Frame 4: TFL=20

RAM
58.3.9 Reception procedure

Before starting any frame reception, the user must activate the SWP (refer to Section 58.3.3: SWP initialization and activation).

Once SWPACT bit is set in the SWPACT register, a RESUME from slave state sets the SRF flag in the SWPMI_ISR register and automatically enables the SWPMI for the frame reception.

If the SWP bus is already in the ACTIVATED state (for example because a frame transmission is ongoing), the SWPMI core does not need any RESUME by slave state, and the reception can take place immediately.

There are several possible software implementations for a frame reception:
- No software buffer mode,
- Single software buffer mode,
- Multi software buffer mode.

The software buffer usage requires the use of a DMA channel to transfer data from the receive data register in the SWPMI peripheral to the software buffer in the RAM memory.

No software buffer mode

This mode does not require the use of DMA. The SWP frame reception handling is done by polling status flags in the main loop or inside the SWPMI interrupt routine. There is a 32-bit receive data register (SWPMI_RDR) in the SWPMI, allowing to receive up to 4 bytes before reading this register.

The No software buffer mode is selected by resetting RXDMA bit in the SWPMI_CR register.

Once a Start of frame (SOF) is received, the following bytes (payload) are stored in the SWPMI_RDR register. Once the SWPMI_RDR is full, the RXNE flag is set in SWPMI_ISR and an interrupt is generated if RIE bit is set in SWPMI_IER register. The user can read the SWPMI_RDR register and the RXNE flag is cleared automatically when the software is reading the SWPMI_RDR register.

Once the complete frame has been received, including the CRC and the End of frame (EOF), both RXNE and RXBFF flags are set in the SWPMI_ISR register. The user must read the last byte(s) of the payload in the SWPMI_RDR register and set CRXBFF flag in SWPMI_ICR in order to clear the RXBFF flag. The number of data bytes in the payload is available in the SWPMI_RFL register. Again, the RXNE flag is reset automatically when the software is reading the SWPMI_RDR register (refer to Figure 657: SWPMI No software buffer mode reception).

Reading the SWPMI_RDR register while RXNE is cleared will return 0.
Single software buffer mode

This mode allows to receive a complete SWP frame without any CPU intervention using the DMA. The DMA transfers received data from the 32-bit SWPMI_RDR register to the RAM memory, and the software can poll the end of the frame reception using the SWPMI_RBFF flag.

The Single software buffer mode is selected by setting RXDMA bit and clearing RXMODE bit in the SWPMI_CR register.

The DMA must be configured as follows:

The DMA channel or stream must be configured in following mode (refer to DMA section):

- memory to memory mode disabled,
- memory increment mode enabled,
- memory size set to 32-bit,
- peripheral size set to 32-bit,
- peripheral increment mode disabled,
- circular mode disabled,
- data transfer direction set to read from peripheral,
- the number of words to be transferred must be set to 8,
- the source address is the SWPMI_RDR register,
- the destination address is the SWP frame buffer in RAM.

Then the user must:

1. Set RXDMA bit in the SWPMI_CR register
2. Set RXBFIE bit in the SWPMI_IER register
3. Enable stream or channel in DMA module.

A DMA request is issued by SWPMI when RXNE flag is set in SWPMI_ISR. The RXNE flag is cleared automatically when the DMA is reading the SWPMI_RDR register.

In the SWPMI interrupt routine, the user must check RXBFF bit in the SWPMI_ISR register. If it is set, the user must:
1. Disable stream or channel in DMA module
2. Read the number of bytes in the received frame payload in the SWPMI_RFL register
3. Read the frame payload in the RAM buffer
4. Enable stream or channel in DMA module
5. Set CRXBFF bit in the SWPMI_ICR register to clear RXBFF flag (refer to Figure 658: SWPMI single software buffer mode reception).

**Figure 658. SWPMI single software buffer mode reception**

<table>
<thead>
<tr>
<th>SWP input</th>
<th>SOF</th>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
<th>D8</th>
<th>D9</th>
<th>D10</th>
<th>D11</th>
<th>D12</th>
<th>D13</th>
<th>CRC</th>
<th>EOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal buffer</td>
<td>****</td>
<td>D3-02-01-00</td>
<td>D7-06-05-04</td>
<td>D11-010-09-08</td>
<td>x=x13-012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SWPMI_RDR</td>
<td>****</td>
<td>D3-02-01-00</td>
<td>D7-06-05-04</td>
<td>D11-010-09-08</td>
<td>x=x13-012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SWPMI_RFL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXNE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXBFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUSPEND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Multi software buffer mode

This mode allows to work with several frame buffers in the RAM memory, in order to ensure a continuous reception, keeping a very low CPU load, using the DMA. The frame payloads are stored in the RAM memory, together with the frame status flags. The software can check the DMA counters and status flags at any time to handle the received SWP frames in the RAM memory.

The Multi software buffer mode must be used in combination with the DMA in circular mode.

The Multi software buffer mode is selected by setting both RXDMA and RXMODE bits in SWPMI_CR register.
In order to work with n reception buffers in RAM, the DMA channel or stream must be configured in following mode (refer to DMA section):

- memory to memory mode disabled,
- memory increment mode enabled,
- memory size set to 32-bit,
- peripheral size set to 32-bit,
- peripheral increment mode disabled,
- circular mode enabled,
- data transfer direction set to read from peripheral,
- the number of words to be transferred must be set to 8 x n (8 words per buffer),
- the source address is the SWPMI_TDR register,
- the destination address is the buffer1 address in RAM

Then the user must:
1. Set RXDMA in the SWPMI_CR register
2. Set RXBFIE in the SWPMI_IER register
3. Enable stream or channel in the DMA module.

In the SWPMI interrupt routine, the user must check RXBFF in the SWPMI_ISR register. If it is set, the user must set CRXBFF bit in the SWPMI_ICR register to clear RXBFF flag and the user can read the first frame payload received in the first buffer (at the RAM address set in DMA2_CMAR1).

The number of data bytes in the payload is available in bits [23:16] of the last 8th word.

In the next SWPMI interrupt routine occurrence, the user will read the second frame received in the second buffer (address set in DMA2_CMAR1 + 8), and so on (refer to Figure 659: SWPMI Multi software buffer mode reception).

In case the application software cannot ensure to handle the SMPMI interrupt before the next frame reception, each buffer status is available in the most significant byte of the 8th buffer word:

- The CRC error flag (equivalent to RXBERF flag in the SWPMI_ISR register) is available in bit 24 of the 8th word. Refer to Section 58.3.10: Error management for an CRC error description.
- The receive overrun flag (equivalent to RXOVRF flag in the SWPMI_ISR register) is available in bit 25 of the 8th word. Refer to Section 58.3.10: Error management for an overrun error description.
- The receive buffer full flag (equivalent to RXBFF flag in the SWPMI_ISR register) is available in bit 26 of the 8th word.

In case of a CRC error, both RXBFF and RXBERF flags are set, thus bit 24 and bit 26 are set.

In case of an overrun, an overrun flag is set, thus bit 25 is set. The receive buffer full flag is set only in case of an overrun during the last word reception; then, both bit 25 and bit 26 are set for the current and the next frame reception.

The software can also read the DMA counter (number of data to transfer) in the DMA registers in order to retrieve the frame which has already been received and transferred into the RAM memory through DMA. For example, if the software works with 4 reception buffers,
and if the DMA counter equals 17, it means that two buffers are ready for reading in the RAM area.

In Multi software buffer reception mode, if the software is reading bits 24, 25 and 26 of the 8th word, it does not need to clear RXBERF, RXOVRF and RXBFF flags after each frame reception.

### Error management

#### Underrun during payload transmission

During the transmission of the frame payload, a transmit underrun is indicated by the TXUNRNF flag in the SWPMI_ISR register. An interrupt is generated if TXBUNREIE bit is set in the SWPMI_IER register.

If a transmit underrun occurs, the SWPMI stops the payload transmission and sends a corrupted CRC (the first bit of the first CRC byte sent is inverted), followed by an EOF. If DMA is used, TXDMA bit in the SWPMI_CR register is automatically cleared.
Any further write to the SWPMI_TDR register while TXUNRF is set will be ignored. The user must set CTXUNRF bit in the SWPMI_ICR register to clear TXUNRF flag.

**Overrun during payload reception**

During the reception of the frame payload, a receive overrun is indicated by RXOVRF flag in the SWPMI_ISR register. If a receive overrun occurs, the SWPMI does not update SWPMI_RDR with the incoming data. The incoming data will be lost.

The reception carries on up to the EOF and, if the overrun condition disappears, the RXBFF flag is set. When RXBFF flag is set, the user can check the RXOVRF flag. The user must set CRXOVRF bit in the SWPMI_ICR register to clear RXBOVRF flag.

If the user wants to detect the overrun immediately, RXBOVREIE bit in the SWPMI_IER register can be set in order to generate an interrupt as soon as the overrun occurs.

The RXOVRF flag is set at the same time as the RXNE flag, two SWPMI_RDR reads after the overrun event occurred. It indicates that at least one received byte was lost, and the loaded word in SWPMI_RDR contains the bytes received just before the overrun.

In Multi software buffer mode, if RXOVRF flag is set for the last word of the received frame, then the overrun bit (bit 25 of the 8th word) is set for both the current and the next frame.

**CRC error during payload reception**

Once the two CRC bytes have been received, if the CRC is wrong, the RXBERF flag in the SWPMI_ISR register is set after the EOF reception. An interrupt is generated if RXBEIE bit in the SWPMI_IER register is set (refer to Figure 660: SWPMI single buffer mode reception with CRC error). The user must set CRXBERF bit in SWPMI_ICR to clear RXBERF flag.

**Missing or corrupted stuffing bit during payload reception**

When a stuffing bit is missing or is corrupted in the payload, RXBERF and RXBFF flags are set in SWPMI_ISR after the EOF reception.
Corrupted EOF reception

Once an SOF has been received, the SWPMI accumulates the received bytes until the reception of an EOF (ignoring any possible SOF). Once an EOF has been received, the SWPMI is ready to start a new frame reception and waits for an SOF.

In case of a corrupted EOF, RXBERF and RXBFF flags will be set in the SWPMI_ISR register after the next EOF reception.

*Note:* In case of a corrupted EOF reception, the payload reception carries on, thus the number of bytes in the payload might get the value 31 if the number of received bytes is greater than 30. The number of bytes in the payload is read in the SWPMI_RFL register or in bits [23:16] of the 8th word of the buffer in the RAM memory, depending on the operating mode.

58.3.11 Loopback mode

The loopback mode can be used for test purposes. The user must set LPBK bit in the SWPMI_CR register in order to enable the loopback mode.

When the loopback mode is enabled, SWPMI_TX and SWPMI_RX signals are connected together. As a consequence, all frames sent by the SWPMI will be received back.

58.4 SWPMI low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. SWPMI interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>A RESUME from SUSPENDED mode issued by the slave can wake up the device from Stop mode if the swpmi_ker_ck is HSI (refer to Section 58.3.1: SWPMI block diagram).</td>
</tr>
<tr>
<td>Standby</td>
<td>The SWPMI is stopped.</td>
</tr>
</tbody>
</table>
58.5 **SWPMI interrupts**

All SWPMI interrupts are connected to the NVIC.

To enable the SWPMI interrupt, the following sequence is required:
1. Configure and enable the SWPMI interrupt channel in the NVIC
2. Configure the SWPMI to generate SWPMI interrupts (refer to the SWPMI_IER register).

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Exit the Sleep mode</th>
<th>Exit the Stop mode</th>
<th>Exit the Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive buffer full</td>
<td>RXBFF</td>
<td>RXBFIE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Transmit buffer empty</td>
<td>TXBETF</td>
<td>TXBEIE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Receive buffer error (CRC error)</td>
<td>RXBERF</td>
<td>RXBEIE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Receive buffer overrun</td>
<td>RXOVRF</td>
<td>RXBOVEREIE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Transmit buffer underrun</td>
<td>TXUNRF</td>
<td>TXBUNREIE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Receive data register not empty</td>
<td>RXNE</td>
<td>RIE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Transmit data register full</td>
<td>TXE</td>
<td>TIE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Transfer complete flag</td>
<td>TCF</td>
<td>TCIE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Slave resume flag</td>
<td>SRF</td>
<td>SRIE</td>
<td>yes</td>
<td>yes (1)</td>
<td>no</td>
</tr>
<tr>
<td>Transceiver ready flag</td>
<td>RDYF</td>
<td>RDYIE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

1. If HSI is selected for swpmi_ker_ck.
58.6 **SWPMI registers**

Refer to [Section 1.2](#) of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

### 58.6.1 SWPMI Configuration/Control register (SWPMI_CR)

Address offset: 0x00

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-20</th>
<th>Bit 19-10</th>
<th>Bit 9-8</th>
<th>Bit 7-6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29-22</th>
<th>Bit 21-16</th>
<th>Bit 15-12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9-6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>SWPEN</td>
<td>DEACT</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWPTEN**: Single wire protocol master transceiver enable

This bit is used to enable the transceiver and control the SWPMI_IO with SWPMI (refer to [Section 58.3.3: SWP initialization and activation](#)).

0: SWPMI_IO pin is controlled by GPIO controller
1: SWPMI_IO transceiver is controlled by SWPMI

Bit 10 **DEACT**: Single wire protocol master interface deactivate

This bit is used to request the SWP DEACTIVATED state. Setting this bit has the same effect as clearing the SWPACT, except that a possible incoming RESUME by slave will keep the SWP in the ACTIVATED state.

Bits 9:6 Reserved, must be kept at reset value.

Bit 5 **SWPACT**: Single wire protocol master interface activate

This bit is used to activate the SWP bus (refer to [Section 58.3.3: SWP initialization and activation](#)).

0: SWPMI_IO is pulled down to ground, SWP bus is switched to DEACTIVATED state
1: SWPMI_IO is released, SWP bus is switched to SUSPENDED state

To be able to set SWPACT bit, DEACT bit must be have been cleared previously.

Bit 4 **LPBK**: Loopback mode enable

This bit is used to enable the loopback mode

0: Loopback mode is disabled
1: Loopback mode is enabled

*Note: This bit cannot be written while SWPACT bit is set.*

Bit 3 **TXMODE**: Transmission buffering mode

This bit is used to choose the transmission buffering mode. This bit is relevant only when TXDMA bit is set (refer to [Table 426: Buffer modes selection for transmission/reception](#)).

0: SWPMI is configured in Single software buffer mode for transmission
1: SWPMI is configured in Multi software buffer mode for transmission

*Note: This bit cannot be written while SWPACT bit is set.*
Bit 2 **RXMODE**: Reception buffering mode

This bit is used to choose the reception buffering mode. This bit is relevant only when TXDMA bit is set (refer to Table 426: Buffer modes selection for transmission/reception).

0: SWPMI is configured in Single software buffer mode for reception
1: SWPMI is configured in Multi software buffer mode for reception.

*Note:* This bit cannot be written while SWPACT bit is set.

Bit 1 **TXDMA**: Transmission DMA enable

This bit is used to enable the DMA mode in transmission

0: DMA is disabled for transmission
1: DMA is enabled for transmission

*Note:* TXDMA is automatically cleared if the payload size of the transmitted frame is given as 0x00 (in the least significant byte of TDR for the first word of a frame). TXDMA is also automatically cleared on underrun events (when TXUNRF flag is set in the SWP_ISR register)

Bit 0 **RXDMA**: Reception DMA enable

This bit is used to enable the DMA mode in reception

0: DMA is disabled for reception
1: DMA is enabled for reception

Table 426. Buffer modes selection for transmission/reception

<table>
<thead>
<tr>
<th>Buffer mode</th>
<th>No software buffer</th>
<th>Single software buffer</th>
<th>Multi software buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXMODE/TXMODE</td>
<td>x</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>RXDMA/TXDMA</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

58.6.2 **SWPMI Bitrate register (SWPMI_BRR)**

Address offset: 0x04

Reset value: 0x0000 0001

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

Bits 7:0 **BR[7:0]**: Bitrate prescaler

This field must be programmed to set SWP bus bitrate, taking into account the \( F_{swpmi\_ker\_ck} \) programmed in the RCC (Reset and Clock Control), according to the following formula:

\[ F_{SWP} = F_{swpmi\_ker\_ck} / ((BR[7:0]+1)\times4) \]

*Note:* The programmed bit rate must stay within the following range: from 100 kbit/s up to 2 Mbit/s.

**BR[7:0]** cannot be written while SWPACT bit is set in the SWPMI_CR register.
### SWPMI Interrupt and Status register (SWPMI_ISR)

Address offset: 0x0C  
Reset value: 0x0000 02C2

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

Bits 31:12  Reserved, must be kept at reset value.

**Bit 11 RDYF: transceiver ready flag**

This bit is set by hardware as soon as transceiver is ready. After setting the SWPTEN bit in SWPMI_CR register to enable the SWPMI IO transceiver, software must wait for this flag to be set before setting the SWPACT bit to activate the SWP bus.

0: transceiver not ready  
1: transceiver ready

**Bit 10 DEACTF: DEACTIVATED flag**

This bit is a status flag, acknowledging the request to enter the DEACTIVATED mode.

0: SWP bus is in ACTIVATED or SUSPENDED state  
1: SWP bus is in DEACTIVATED state

If a RESUME by slave state is detected by the SWPMI while DEACT bit is set by software, the SRF flag will be set, DEACTF will not be set and SWP will move in ACTIVATED state.

**Bit 9 SUSP: SUSPEND flag**

This bit is a status flag, reporting the SWP bus state

0: SWP bus is in ACTIVATED state  
1: SWP bus is in SUSPENDED or DEACTIVATED state

**Bit 8 SRF: Slave resume flag**

This bit is set by hardware to indicate a RESUME by slave detection. It is cleared by software, writing 1 to CSRF bit in the SWPMI_ICR register.

0: No Resume by slave state detected  
1: A Resume by slave state has been detected during the SWP bus SUSPENDED state

**Bit 7 TCF: Transfer complete flag**

This flag is set by hardware as soon as both transmission and reception are completed and SWP is switched to the SUSPENDED state. It is cleared by software, writing 1 to CTCF bit in the SWPMI_ICR register.

0: Transmission or reception is not completed  
1: Both transmission and reception are completed and SWP is switched to the SUSPENDED state

**Bit 6 TXE: Transmit data register empty**

This flag indicates the transmit data register status

0: Data written in transmit data register SWPMI_TDR is not transmitted yet  
1: Data written in transmit data register SWPMI_TDR has been transmitted and SWPMI_TDR can be written to again
Bit 5 **RXNE**: Receive data register not empty  
This flag indicates the receive data register status  
0: Data is not received in the SWPMI_RDR register  
1: Received data is ready to be read in the SWPMI_RDR register

Bit 4 **TXUNRF**: Transmit underrun error flag  
This flag is set by hardware to indicate an underrun during the payload transmission i.e. SWPMI_TDR has not been written in time by the software or the DMA. It is cleared by software, writing 1 to the CTXUNRF bit in the SWPMI_ICR register.  
0: No underrun error in transmission  
1: Underrun error in transmission detected

Bit 3 **RXOVRF**: Receive overrun error flag  
This flag is set by hardware to indicate an overrun during the payload reception, i.e. SWPMI_RDR has not be read in time by the software or the DMA. It is cleared by software, writing 1 to CRXOVRF bit in the SWPMI_ICR register.  
0: No overrun in reception  
1: Overrun in reception detected

Bit 2 **RXBERF**: Receive CRC error flag  
This flag is set by hardware to indicate a CRC error in the received frame. It is set synchronously with RXBFF flag. It is cleared by software, writing 1 to CRXBERF bit in the SWPMI_ICR register.  
0: No CRC error in reception  
1: CRC error in reception detected

Bit 1 **TXBEF**: Transmit buffer empty flag  
This flag is set by hardware to indicate that no more SWPMI_TDR update is required to complete the current frame transmission. It is cleared by software, writing 1 to CTXBEF bit in the SWPMI_ICR register.  
0: Frame transmission buffer no yet emptied  
1: Frame transmission buffer has been emptied

Bit 0 **RXBFF**: Receive buffer full flag  
This flag is set by hardware when the final word for the frame under reception is available in SWPMI_RDR. It is cleared by software, writing 1 to CRXBFF bit in the SWPMI_ICR register.  
0: The last word of the frame under reception has not yet arrived in SWPMI_RDR  
1: The last word of the frame under reception has arrived in SWPMI_RDR

### 58.6.4 SWPMI Interrupt Flag Clear register (SWPMI_ICR)

Address offset: 0x10  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RXNE</td>
<td>0</td>
<td>RXOVRF</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>0</td>
<td>RXBERF</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>0</td>
<td>TXUNRF</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>0</td>
<td>TXBEF</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>0</td>
<td>RXBFF</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>0</td>
<td>CRDY</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>0</td>
<td>CSRF</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>0</td>
<td>CTF C</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>0</td>
<td>CTXUN</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>0</td>
<td>CRXOV</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>0</td>
<td>CRXBER</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>0</td>
<td>CTXBE</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>0</td>
<td>CRXB</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>0</td>
<td>CRXBF</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>0</td>
<td>RC_W1</td>
<td>0</td>
</tr>
</tbody>
</table>

RC0455 Rev 3  & 2295/2974
58.6.5 SWPMI Interrupt Enable register (SMPMI_IER)

Address offset: 0x14
Reset value: 0x0000 0000
Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RDYIE**: Transceiver ready interrupt enable

- 0: Interrupt is inhibited
- 1: A SWPMI interrupt is generated whenever RDYF flag is set in the SWPMI_ISR register

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SRIE**: Slave resume interrupt enable

- 0: Interrupt is inhibited
- 1: An SWPMI interrupt is generated whenever SRF flag is set in the SWPMI_ISR register

Bit 7 **TCIE**: Transmit complete interrupt enable

- 0: Interrupt is inhibited
- 1: An SWPMI interrupt is generated whenever TCF flag is set in the SWPMI_ISR register

Bit 6 **TIE**: Transmit interrupt enable

- 0: Interrupt is inhibited
- 1: An SWPMI interrupt is generated whenever TXE flag is set in the SWPMI_ISR register

Bit 5 **RIE**: Receive interrupt enable

- 0: Interrupt is inhibited
- 1: An SWPMI interrupt is generated whenever RXNE flag is set in the SWPMI_ISR register

Bit 4 **TXUNRIE**: Transmit underrun error interrupt enable

- 0: Interrupt is inhibited
- 1: An SWPMI interrupt is generated whenever TXBUNRF flag is set in the SWPMI_ISR register

Bit 3 **RXOVRIE**: Receive overrun error interrupt enable

- 0: Interrupt is inhibited
- 1: An SWPMI interrupt is generated whenever RXBOVRF flag is set in the SWPMI_ISR register

Bit 2 **RXBERIE**: Receive CRC error interrupt enable

- 0: Interrupt is inhibited
- 1: An SWPMI interrupt is generated whenever RXBERF flag is set in the SWPMI_ISR register

Bit 1 **TXBEIE**: Transmit buffer empty interrupt enable

- 0: Interrupt is inhibited
- 1: An SWPMI interrupt is generated whenever TXBEF flag is set in the SWPMI_ISR register

Bit 0 **RXBFIE**: Receive buffer full interrupt enable

- 0: Interrupt is inhibited
- 1: An SWPMI interrupt is generated whenever RXBFF flag is set in the SWPMI_ISR register
58.6.6 SWPMI Receive Frame Length register (SWPMI_RFL)

Address offset: 0x18
Reset value: 0x0000 0000

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **RFL[4:0]**: Receive frame length

RFL[4:0] is the number of data bytes in the payload of the received frame. The two least significant bits RFL[1:0] give the number of relevant bytes for the last SWPMI_RDR register read.

58.6.7 SWPMI Transmit data register (SWPMI_TDR)

Address offset: 0x1C
Reset value: 0x0000 0000

Bits 31:0 **TD[31:0]**: Transmit data

Contains the data to be transmitted. Writing to this register triggers the SOF transmission or the next payload data transmission, and clears the TXE flag.

58.6.8 SWPMI Receive data register (SWPMI_RDR)

Address offset: 0x20
Reset value: 0x0000 0000
Bits 31:0  **RD[31:0]**: received data  
Contains the received data  
Reading this register is clearing the RXNE flag.

### 58.6.9 SWPMI Option register (SWPMI_OR)

Address offset: 0x24  
Reset value: 0x0000 0000

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

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

Bit 1  **SWP_CLASS**: SWP class selection  
This bit is used to select the SWP class (refer to Section 58.3.3: SWP initialization and activation).

0: Class C: SWPMI_IO uses directly VDD voltage to operate in class C.  
This configuration must be selected when VDD is in the range [1.62 V to 1.98 V]  
1: Class B: SWPMI_IO uses an internal voltage regulator to operate in class B.  
This configuration must be selected when VDD is in the range [2.70 V to 3.30 V]

Bit 0  **SWP_TBYP**: SWP transceiver bypass  
This bit is used to bypass the internal transceiver (SWPMI_IO), and connect an external transceiver.

0: Internal transceiver is enabled. The external interface for SWPMI is SWPMI_IO (SWPMI_RX, SWPMI_TX and SWPMI_SUSPEND signals are not available on GPIOs)  
1: Internal transceiver is disabled. SWPMI_RX, SWPMI_TX and SWPMI_SUSPEND signals are available as alternate function on GPIOs. This configuration is selected to connect an external transceiver
## 58.6.10 SWPMI register map and reset value table

#### Table 427. SWPMI register map and reset values

| Offset | Register name   | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | SWPMI_CR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | SWPMI_BRR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | RESERVED       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | SWPMI_ISR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 1  | 0  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | SWPMI_ICR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x14   | SWPMI_IER      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x18   | SWPMI_RFL      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x1C   | SWPMI_TDR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TD[31:0]       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x20   | SWPMI_RDR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | RD[31:0]       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x24   | SWPMI_OR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.3 on page 128 for the register boundary addresses.
59  Management data input/output (MDIOS)

59.1  MDIOS introduction

An MDIO bus can be useful in systems where a master chip needs to manage (configure and get status data from) one or multiple slave chips. The bus protocol uses only two signals:

- MDC: the management data clock
- MDIO: the data line carrying the opcode (write or read), the slave (port) address, the MDIOS register address, and the data

In each transaction, the master either reads the contents of an MDIOS register in one of its slaves, or it writes data to an MDIOS register in one of its slaves.

The MDIOS peripheral serves as a slave interface to an MDIO bus. An MDIO master can use the MDC/MDIO lines to write and read 32 16-bit MDIOS registers which are held in the MDIOS. These MDIOS registers are managed by the firmware, thus allowing the MDIO master to configure the application running on the STM32 and get status information from it.

The MDIOS can operate in Stop mode, optionally waking up the STM32 if the MDIO master performs a read or a write to one of its MDIOS registers.

59.2  MDIOS main features

The MDIOS includes the following features:

- 32 MDIOS registers addresses, each of which is managed using separate input and output data registers:
  - 32 x 16-bit firmware read/write, MDIOS read-only output data registers
  - 32 x 16-bit firmware read-only, MDIOS write-only input data registers
- Configurable slave (port) address
- Independently maskable interrupts/events:
  - MDIOS register write
  - MDIOS register read
  - MDIOS protocol error
- Able to operate in and wake up from Stop mode
59.3 MDIOS functional description

59.3.1 MDIOS block diagram

![MDIOS block diagram](image)

**59.3.2 MDIOS pins and internal signals**

*Table 428* lists the MDIOS inputs and output signals connected to package pins or balls, while *Table 429* shows the internal PWR signals.

**Table 428. MDIOS input/output signals connected to package pins or balls**

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDIOS_MDC</td>
<td>Digital input</td>
<td>MDIO master clock</td>
</tr>
<tr>
<td>MDIOS_MDIO</td>
<td>Digital input/output</td>
<td>MDIO signal (opcode, address, input/output data)</td>
</tr>
</tbody>
</table>

**Table 429. MDIOS internal input/output signals**

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mdios_wkup</td>
<td>Digital output</td>
<td>MDIOS wakeup signal</td>
</tr>
<tr>
<td>mdios_it</td>
<td>Digital output</td>
<td>MDIOS interrupt signal</td>
</tr>
<tr>
<td>mdios_pclk</td>
<td>Digital input</td>
<td>APB clock</td>
</tr>
</tbody>
</table>
59.3.3 MDIOS protocol

The MDIOS protocol uses two signals:
1. MDIOS_MDC: the clock, always driven by the master
2. MDIOS_MDIO: signal carrying the opcode, address, and bidirectional data

Each transaction is performed using a "frame". Each frame contains 32 bits: 14 control bits, 2 turn-around bits, and then 16 data bits, each passed serially.

- 14 control bits, driven by the master
  - 2 start bits: always "01"
  - 2 opcode bits: read="10", write="01"
  - 5 port address bits, indicating which slave device is being addressed
  - 5 MDIOS register address bits, up to 32 MDIOS registers can be addressed in each slave

- 2 turn-around state bits
  - On write operations, the master drives “10”
  - On read operations, the first bit is high-impedance, and the second bit is driven by the slave to ‘0’

- 16 data bits
  - On write operations, data written to slave’s MDIOS register is driven by the master
  - On read operations, data read from slave’s MDIOS register is driven by the slave

Each frame is usually preceded by a preamble, where the MDIOS stays at ‘1’ for 32 MDC clocks. The master can continue to keep MDIO at ‘1’, indicating the “idle” condition, when it has no frame to send.

When MDIO signal is driven by the master, MDIOS samples it using the rising edge of MDC. When MDIOS drives MDIO, the output changes on the rising edge of MDC.
59.3.4 MDIOS enabling and disabling

The MDIOS is enabled by setting the EN bit in the MDIOS_CR register. When EN=1, the MDIOS monitors the MDIO bus and service frames addressed to one of its MDIOS registers.

When the MDIOS is enabled (setting EN to ‘1’), the same write operation to the MDIOS_CR register must properly set the PORT_ADDRESS[4:0] field to indicate the slave port address. A frame is ignored by the MDIOS if its port address is not the same as PORT_ADDRESS[4:0] (presumably intended for another slave).

When EN=0, the MDIOS ignores the frames being transmitted on the MDC/MDIO lines, and the IP is in a reduced consumption mode. Clearing EN also clears all of the DIN registers. If EN is cleared while the MDIOS is receiving a frame, the frame is aborted and the data is lost.

When the MDIOS is enabled, then disabled and subsequently re-enabled, the status flags are not cleared. For a correct operation the firmware shall clear the status flag before re-enabling the MDIOS.

59.3.5 MDIOS data

From the point of view of the MDIO master, there are 32 16-bit MDIOS registers in the MDIOS which can be written and read. In reality, for each MDIOS register ‘n’ there are two sets of registers: DINn[15:0] and DOUTn[15:0].

Input data

When the MDIO master transmits a frame which writes to MDIOS register ‘n’ in the MDIOS, it is the DINn[15:0] register which is updated with the incoming data. The DIN registers (DIN0 - DIN31) can be read by the firmware, but they can be written only by the MDIO master via the MDIO bus.

The contents of DINn change immediately after the MDC rising edge when the last data bit is sampled.

If the firmware happens to read the contents of DINn at the moment that it is being updated, there is a possibility that the value read is corrupted (a bit-by-bit cross between the old value and the new value). For this reason, the firmware must assure that two subsequent reads from the same DINn register give the same value and assure that the data is stable when it is read. In the very worst case, the firmware needs to read DINn four times: first to get the old value, second to get an incoherent value (when reading at the moment the register changes), third to get the new value, and forth to confirm the new value.

If the firmware uses the WRF interrupt and can guarantee that it reads the DINn register before any new MDIOS write frame completes, the firmware can perform a single read.

If the MDIO master performs a write operation with a register address that is greater than 31, the MDIOS ignores the frame (the data is not saved and no flag is set).
Output data

When the MDIOS receives a frame which requests to read register ‘n’, it returns the value found in the DOUTn[15:0] register. Thus, if the MDIO master expects to read the same value which it previously wrote to MDIOS register ‘n’, the firmware must copy the data from DINn to DOUTn each time new data is written to DINn. For correct operation, the firmware must copy the data to the DOUTn register within a preamble (if the master sends preambles before each frame) plus 15 cycles time.

When an MDIOS register is read via the MDIO bus, the MDIOS passes the 16-bit value (from the corresponding DOUTn register) to the MDIOS clock domain during the 15th cycle of the read frame. If the firmware attempts to write the DOUTn register while the MDIO Master is currently reading MDIOS register ‘n’, then the firmware write operation is ignored if it occurs during the 15th cycle of the frame (during a one-MDC-cycle window). Therefore, **after writing a DOUTn register, the firmware must read back the same DOUTn register and confirm that the value was actually written.** If the DOUTn register does not contain the value which was written, then the firmware can simply try writing and re-reading again.

If the MDIOS frequency is very slow compared to the mdios_pclk frequency, then it might be best not to tie up the CPU by continuously writing and re-reading a DOUT register. Please note that the read flag (RDFn) is set as soon as the DOUTn value is passed to the MDIOS clock domain. Thus, when a write to DOUTn is ignored (when the value read back is not the value which was just written), then the firmware can use a read interrupt to know when it is able to write DOUTn.

Here is a procedure which can be used if the MDC clock is very slow:

1. Write DOUTn.
2. Assure that all of the read flags are zero (MDIOS_RDFR = 0x0000). Clear the flags if necessary using MDIOS_CRDFR.
3. Read back the same DOUTn register and compare the value with the value which was written in step 1.
4. If the values are the same, then the procedure is done. Otherwise, continue to step 5.
5. Enable read interrupts by setting the RDIE bit in MDIOS_CR1.
6. In the interrupt routine, assure that RDFn is set. (no other read flags are set before bit n).
7. There is a 31 cycle + preamble time window (if the master sends a preamble before each frame) to write DOUTn safely without doing a read-back and compare. If this maximum delay cannot be guaranteed, go back to step #1.

If the MDIO master performs a read operation with a register address which is greater than 31, the MDIOS returns a data value of all zeros.
59.3.6 MDIOS APB frequency
Whenever the firmware reads from an MDIOS_DINRn register or writes to an MDIOS_DOUTRn register, the frequency of the APB bus must be at least 1.5 times the MDC frequency. For example, if MDC is at 20MHz, the APB must be at 30MHz or faster.

59.3.7 Write/read flags and interrupts
When MDIOS register ‘n’ is written via the MDIO bus, the WRFn bit in the MDIOS_WRFR register is set. WRFn becomes ‘1’ at the moment that DINn is updated, which is when the last data bit is sampled on a write frame. An interrupt is generated if WRIEN=1 (in the MDIOS_CR register). WRFn is cleared by software by writing ‘1’ to CWRFn (in the MDIOS_CWRFR register).

When MDIOS register ‘n’ is read via the MDIO bus, the RDFn bit in the MDIOS_RDFR register is set. RDFn becomes ‘1’ at the moment that DOUTn is copied to the MDC clock domain, which is on the 15th cycle of a read frame. An interrupt is generated if RDIEN=1 (in the MDIOS_CR register). RDFn is cleared by software by writing ‘1’ to CRDFn (in the MDIOS_CRDFR register).

59.3.8 MDIOS error management
There are three types of errors with their corresponding error flags:
- Preamble error: PERF (bit 0 of MDIOS_SR register)
- Start error: SERF (bit 1 of MDIOS_SR register)
- Turnaround error: TERF (bit 2 of MDIOS_SR register)

Each error flag is set by hardware when the corresponding error condition occurs. Each flag can be cleared by writing ‘1’ to the corresponding bit in the clear flag register (MDIOS_CLRFR).

An interrupt occurs if any of the three error flags is set while EIE=1 (MDIOS_CR).

Besides setting an error flag, the MDIOS performs no action for a frame in which an error is detected: the DINn registers are not updated and the MDIO line is not forced during the data phase.

For a given frame, errors do not accumulate. For example, if a preamble error is detected, no check is done for a start error or a turnaround error for the rest of the current frame.

When DPC=0, following an detected error, all new frames and errors are ignored until a complete full preamble has been detected.

When DPC=1 (Disable Preamble Check, MDIOS_CR[7]), all frames and new errors are ignored as long as one of the error flags is set. As soon as the error bit is cleared, the MDIOS starts looking for a start sequence. Thus, the application must clear the error flag only when it is sure that no frame is currently in progress. Otherwise, the MDIOS likely misinterprets the bits being sent and becomes desynchronized with the master.

Preamble errors
A preamble error occurs when a start sequence begins (with MDIO sampled at ‘0’) without being immediately preceded by a preamble (MDIO sampled at ‘1’ for at least 32 consecutive clocks).
Preamble errors are not reported after the MDIOS is first enabled (EN=1 in MDIOS_CR) until after a full preamble is received. This is to avoid an error condition when the peripheral frame detection is enabled while a preamble or frame is already in progress. In this case, the MDIOS ignores the first frame (since it did not first detect a full preamble), but does not set PERF.

If the DPC bit (Disable Preamble Check, MDIOS_CR[7]) is set, then the MDIO Master can send frames without preceding preambles and no preamble error is signaled. When DPC=1, the application must assure that the master is not in the process of sending a frame at the moment that the MDIOS is enabled (EN is set). Otherwise, the slave might become desynchronized with the master.

**Start errors**

A start error occurs when an illegal start sequence occurs or if an illegal command is given. The start sequence must always be “01”, and the command must be either “01” (write) or “10” (read).

As with preamble errors, start errors are not reported until after a full preamble is received.

**Turnaround errors**

A turnaround error occurs when an error is detected in the turnaround bits of write frames. The 15th bit of the write frame must be ‘1’ and the 16th bit must be ‘0’.

Turnaround errors are only reported after a full preamble is received, there is no start error, the port address in the current frame matches and the register address is in the supported range 0 to 31.

### 59.3.9 MDIOS in Stop mode

The MDIOS can operate in Stop mode, responding to all reads, performing all writes, and causing the STM32 to wakeup from Stop mode on MDIOS interrupts.

### 59.3.10 MDIOS interrupts

There is a single interrupt vector for the three types of interrupts (write, read, and error). Any of these interrupt sources can wake the STM32 up from Stop mode. All interrupt flags need to be cleared in order to clear the interrupt line.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write interrupt</td>
<td>WRF[31:0]</td>
<td>WRIE</td>
</tr>
<tr>
<td>Read interrupt</td>
<td>RDF[31:0]</td>
<td>RDIE</td>
</tr>
<tr>
<td>Error interrupt</td>
<td>PERF (preamble), SERF (start), TERF (turnaround)</td>
<td>EIE</td>
</tr>
</tbody>
</table>

Table 430. Interrupt control bits
59.4 MDIOS registers

59.4.1 MDIOS configuration register (MDIOS_CR)

Address offset: 0x00
Reset value: 0x0000 0000

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

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 PORT_ADDRESS[4:0]: slave’s address.
Can be written only when the peripheral is disabled (EN=0).
If the address given by the MDIO master matches PORT_ADDRESS[4:0], then the MDIOS services the frame. Otherwise the frame is ignored.

Bit 7 DPC: disable preamble check.
0: MDIO Master must give preamble before each frame.
1: MDIO Master can send each frame without a preceding preamble, and the MDIOS does not signal a preamble error.
When this bit is set, the application must be sure that no frame is currently in progress when the MDIOS is enabled. Otherwise, the MDIOS can become desynchronized with the master.
This bit cannot be changed unless EN=0 (though it can be changed at the same time that EN is being set).

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 EIE: error interrupt enable.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if any of the error flags (PERF, SERF, or TERF in the MDIOS_SR register) is set.

Bit 2 RDIE: register read interrupt enable.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if any of the read flags (RDF[31:0] in the MDIOS_RDWR register) is set.

Bit 1 WRIE: register write interrupt enable.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if any of the read flags (WRF[31:0] in the MDIOS_WRFR register) is set.

Bit 0 EN: peripheral enable.
0: MDIOS is disabled
1: MDIOS is enabled and monitoring the MDIO bus (MDC/MDIO)
59.4.2 MDIOS write flag register (MDIOS_WRFR)

Address offset: 0x04

Power-on reset value: 0x0000 0000

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

Bits 31:0 **WRF[31:0]**: write flags for MDIOS registers 0 to 31.

Each bit is set by hardware when the MDIO master performs a write to the corresponding MDIOS register. An interrupt is generated if WRIE (in MDIOS_CR) is set.

Each bit is cleared by software by writing ‘1’ to the corresponding CWRF bit in the MDIOS_CWRFR register.

For WRFn:

- 0: MDIOS register ‘n’ is not written by the MDIO master
- 1: MDIOS register ‘n’ is written by the MDIO master and the data is available in DINn[15:0] in the MDIOS_DINRn register

59.4.3 MDIOS clear write flag register (MDIOS_CWRFR)

Address offset: 0x08

Power-on reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **CWRF[31:0]**: clear the write flag

Writing ‘1’ to CWRFn clears the WRFn bit in the MDIOS_WRF register.
### 59.4.4 MDIOS read flag register (MDIOS_RDFR)

Address offset: 0x0C  
Power-on reset value: 0x0000 0000

| RDF[31:16] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Bits 31:0 **RDF[31:0]**: read flags for MDIOS registers 0 to 31.  
Each bit is set by hardware when the MDIO master performs a read from the corresponding MDIOS register. An interrupt is generated if RDIE (in MDIOS_CR) is set.  
Each bit is cleared by software by writing ‘1’ to the corresponding CRDF bit in the MDIOS_CRFDR register.  
For RDFn:  
0: MDIOS register ‘n’ is not read by the MDIO master  
1: MDIOS register ‘n’ is read by the MDIO master

### 59.4.5 MDIOS clear read flag register (MDIOS_CRFDR)

Address offset: 0x10  
Power-on reset value: 0x0000 0000

| CRDF[31:16] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| CRDF[15:0] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

Bits 31:0 **CRDF[31:0]**: clear the read flag  
Writing ‘1’ to CRDFn clears the RDFn bit in the MDIOS_RDF register.
## 59.4.6 MDIOS status register (MDIOS_SR)

Address offset: 0x14  
Power-on reset value: 0x0000 0000  

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

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

- **Bit 2** TERF: turnaround error flag  
  0: No turnaround error has occurred  
  1: A turnaround error has occurred  
  Writing ‘1’ to CTERF (MDIOS_CLRFR) clears this bits.

- **Bit 1** SERF: start error flag  
  0: No start error has occurred  
  1: A start error has occurred  
  Writing ‘1’ to CSERF (MDIOS_CLRFR) clears this bits.

- **Bit 0** PERF: preamble error flag  
  0: No preamble error has occurred  
  1: A preamble error has occurred  
  Writing ‘1’ to CPERF (MDIOS_CLRFR) clears this bit.  
  This bit is not set if DPC (disable preamble check, MDIOS_CR[7]) is set.

**Note:**  Writes to MDIOS_SR have no effect.
59.4.7 MDIOS clear flag register (MDIOS_CLRFR)

Address offset: 0x18
Power-on reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 2 **CTERF**: clear the turnaround error flag
Writing ‘1’ to this bit clears the TERF flag (MDIOS_SR).
When DPC=’1’ (MDIOS_CR[7]), the TERF flag must be cleared only when there is not a frame already in progress.

Bit 1 **CSERF**: clear the start error flag
Writing ‘1’ to this bit clears the SERF flag (MDIOS_SR).
When DPC=’1’ (MDIOS_CR[7]), the SERF flag must be cleared only when there is not a frame already in progress.

Bit 0 **CPERF**: clear the preamble error flag
Writing ‘1’ to this bit clears the PERF flag (MDIOS_SR).

*Note:* Reading MDIOS_CLRFR returns all zeros.
59.4.8 MDIOS input data register x (MDIOS_DINRx)

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)
Reset value: 0x0000 0000

| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0 | DINn[15:0]: input data received from MDIO Master during write frames
This field written by hardware with the 16-bit data received in a write frame which is addressed to MDIOS register ‘n’.

59.4.9 MDIOS output data register x (MDIOS_DOUTRx)

Address offset: 0x180 + 0x04 * x, (x = 0 to 31)
Reset value: 0x0000 0000

| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0 | DOUTn[15:0]: output data sent to MDIO Master during read frames
This field is written by software. These 16 bits are serially output on the MDIO bus during read frames which address the MDIOS register ‘n’.
59.4.10 MDIOS register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Mask</th>
<th>Port Address[4:0]</th>
<th>DPC</th>
<th>EE</th>
<th>ROE</th>
<th>WRE</th>
<th>EN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>MDIOS_CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>MDIOS_WRFR</td>
<td></td>
<td>WRF[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>MDIOS_CWRFR</td>
<td></td>
<td>CWRF[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>MDIOS_RDFR</td>
<td></td>
<td>RDF[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>MDIOS_CRDFR</td>
<td></td>
<td>CRDF[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>MDIOS_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>MDIOS_CLRF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C - 0xFC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x100</td>
<td>MDIOS_DINR0</td>
<td></td>
<td>DIN0[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x104</td>
<td>MDIOS_DINR1</td>
<td></td>
<td>DIN1[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x17C</td>
<td>MDIOS_DINR31</td>
<td></td>
<td>DIN31[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x180</td>
<td>MDIOS_DOUTR0</td>
<td></td>
<td>DOUT0[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x184</td>
<td>MDIOS_DOUTR1</td>
<td></td>
<td>DOUT1[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1FC</td>
<td>MDIOS_DOUTR31</td>
<td></td>
<td>DOUT31[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
60 Secure digital input/output MultiMediaCard interface (SDMMC)

60.1 SDMMC main features

The SD/SDIO, embedded MultiMediaCard (e•MMC) host interface (SDMMC) provides an interface between the AHB bus and SD memory cards, SDIO cards and e•MMC devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at www.mmca.org, published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at www.sdcard.org.

The SDMMC features include the following:

- Compliance with *Embedded MultiMediaCard System Specification Version 4.51*. Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit (HS200 SDMMC_CK speed limited to maximum allowed I/O speed).
- Full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Full compliance with *SD memory card specifications version 4.1*. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Full compliance with *SDIO card specification version 4.0*. Card support for two different databus modes: 1-bit (default) and 4-bit (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Data transfer up to 208 Mbyte/s for the 8-bit mode (depending maximum allowed I/O speed).
- Data and command output enable signals to control external bidirectional drivers.

The MultiMediaCard/SD bus connects cards to the host.

The current version of the SDMMC supports only one SD/SDIO/e•MMC card at any one time and a stack of e•MMC.

60.2 SDMMC implementation

<table>
<thead>
<tr>
<th>SDMMC modes/features(1)</th>
<th>SDMMC1</th>
<th>SDMMC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable delay (SDR104, HS200)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SDMMC_CKIN</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>SDMMC_CDIR, SDMMC_D0DIR</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>SDMMC_D123DIR</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>MDMA data transfer end</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>MDMA command end</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>MDMA buffer end</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>
60.3 SDMMC bus topology

Communication over the bus is based on command/response and data transfers. The basic transaction on the SD/SDIO/eMMC bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers are done in the following ways:
- Block mode: data block(s) with block size $2^N$ bytes with N in the range 0-14.
- SDIO multibyte mode: single data block with block size range 1-512 bytes
- eMMC Stream mode: continuous data stream

Data transfers to/from eMMC cards are done in data blocks or streams.

![Figure 664. SDMMC “no response” and “no data” operations](image)

![Figure 665. SDMMC (multiple) block read operation](image)

**Note:** The Stop Transmission command is not required at the end of a eMMC multiple block read with predefined block count.
Note: The Stop Transmission command is not required at the end of an eMMC multiple block write with predefined block count.

Note: The SDMMC will not send any data as long as the Busy signal is asserted (SDMMC_D0 pulled low).

Stream data transfer operates only in a 1-bit wide bit bus configuration on SDMMC_D0 in single data rate modes (DS, HS, and SDR).
60.4 SDMMC operation modes

Table 433. SDMMC operation modes SD & SDIO

<table>
<thead>
<tr>
<th>SDIO Bus Speed modes (1)(2)</th>
<th>Max Bus Speed (3) [MByte/s]</th>
<th>Max Clock frequency (4) [MHz]</th>
<th>Signal Voltage [V]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS (Default Speed)</td>
<td>12.5</td>
<td>25</td>
<td>3.3</td>
</tr>
<tr>
<td>HS (High Speed)</td>
<td>25</td>
<td>50</td>
<td>3.3</td>
</tr>
<tr>
<td>SDR12</td>
<td>12.5</td>
<td>25</td>
<td>1.8</td>
</tr>
<tr>
<td>SDR25</td>
<td>25</td>
<td>50</td>
<td>1.8</td>
</tr>
<tr>
<td>DDR50</td>
<td>50</td>
<td>50</td>
<td>1.8</td>
</tr>
<tr>
<td>SDR50</td>
<td>50</td>
<td>100</td>
<td>1.8</td>
</tr>
<tr>
<td>SDR104</td>
<td>104</td>
<td>208</td>
<td>1.8</td>
</tr>
</tbody>
</table>

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. SDIO bus speed with 4bit bus width.
4. Maximum frequency depending on maximum allowed IO speed.

SDR104 mode requires variable delay support using sampling point tuning. The use of variable delay is optional for SDR50 mode.

Table 434. SDMMC operation modes eMMC

<table>
<thead>
<tr>
<th>eMMC bus speed modes (1)(2)</th>
<th>Max bus speed (3) [MByte/s]</th>
<th>Max clock frequency (4) [MHz]</th>
<th>Signal voltage [V]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legacy compatible</td>
<td>26</td>
<td>26</td>
<td>3/1.8/1.2V</td>
</tr>
<tr>
<td>High speed SDR</td>
<td>52</td>
<td>52</td>
<td>3/1.8/1.2V</td>
</tr>
<tr>
<td>High speed DDR</td>
<td>104</td>
<td>52</td>
<td>3/1.8/1.2V</td>
</tr>
<tr>
<td>High speed HS200</td>
<td>200</td>
<td>200</td>
<td>1.8/1.2V</td>
</tr>
</tbody>
</table>

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. eMMC bus speed with 8bit bus width.
4. Maximum frequency depending on maximum allowed IO speed.

HS200 mode requires variable delay support using sampling point tuning.
60.5 SDMMC functional description

The SDMMC consists of four parts:

- The AHB slave interface accesses the SDMMC adapter registers, and generates interrupt signals and IDMA control signals.
- The SDMMC adapter block provides all functions specific to the eMMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The internal DMA (IDMA) block with its AHB master interface.
- A delay block (DLYB) taking care of the receive data sample clock alignment. The delay block is NOT part of the SDMMC. A delay block is mandatory when supporting SDR104 or HS200.

60.5.1 SDMMC block diagram

Figure 669 shows the SDMMC block diagram.

![SDMMC block diagram](MSv41694V2)

60.5.2 SDMMC pins and internal signals

Table 435 lists the SDMMC internal input/output signals, Table 436 the SDMMC pins (alternate functions).

Table 435. SDMMC internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sdmmc_ker_ck</td>
<td>Digital input</td>
<td>SDMMC kernel clock</td>
</tr>
<tr>
<td>sdmmc_hclk</td>
<td>Digital input</td>
<td>AHB clock</td>
</tr>
<tr>
<td>sdmmc_it</td>
<td>Digital output</td>
<td>SDMMC global interrupt</td>
</tr>
<tr>
<td>sdmmc_dataend_trg</td>
<td>Digital output</td>
<td>SDMMC data end trigger for MDMA</td>
</tr>
<tr>
<td>sdmmc_cmdend_trg</td>
<td>Digital output</td>
<td>SDMMC command end trigger for MDMA</td>
</tr>
</tbody>
</table>
The SDMMC_D[7:0] lines have different operating modes:

- By default, SDMMC_D0 line is used for data transfer. After initialization, the host can change the databus width.
- For an eMMC, 1-bit (SDMMC_D0), 4-bit (SDMMC_D[3:0]) or 8-bit (SDMMC_D[7:0]) data bus widths can be used.
- For an SD or an SDIO card, 1-bit (SDMMC_D0) or 4-bit (SDMMC_D[3:0]) can be used. All data lines operate in push-pull mode.

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the data lines is indicated with I/O direction signals. The SDMMC_D0DIR signal indicates the I/O direction for the SDMMC_D0 data line, the SDMMC_D123DIR for the SDMMC_D[3:1] data lines.

SDMMC_CMD only operates in push-pull mode:

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the SDMMC_CMD line is indicated with the I/O direction signal SDMMC_CDIR.
SDMMC_CK clock to the card originates from sdmmc_ker_ck:

- When the sdmmc_ker_ck clock has 50% duty cycle, it can be used even in bypass mode (CLKDIV = 0).
- When the sdmmc_ker_ck duty cycle is not 50%, the CLKDIV must be used to divide it by 2 or more (CLKDIV > 0).
- The phase relation between the SDMMC_CMD / SDMMC_D[7:0] outputs and the SDMMC_CK can be selected through the NEGEDGE bit. The phase relation depends on the CLKDIV, NEGEDGE, and DDR settings. See Figure 670.

Figure 670. SDMMC Command and data phase relation

Table 437. SDMMC Command and data phase selection

<table>
<thead>
<tr>
<th>CLKDIV</th>
<th>DDR</th>
<th>NEGEDGE</th>
<th>SDMMC_CK</th>
<th>Command out</th>
<th>Data out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>x</td>
<td>sdmmc_ker_ck</td>
<td>generated on sdmmc_ker_ck falling edge</td>
<td></td>
</tr>
<tr>
<td>&gt;0</td>
<td>0</td>
<td>0</td>
<td>generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

By default, the sdmmc_io_in_ck feedback clock input is selected for sampling incoming data in the SDMMC receive path. It is derived from the SDMMC_CK pin.

For tuning the phase of the sampling clock to accommodate the receive data timing, the DLYB delay block available on the device can be connected between sdmmc_io_in_ck
signal (DLYB input dlyb_in_ck) and sdmmc_fb_ck clock input of SDMMC (DLYB output dlyb_out_ck). Selecting the sdmmc_fb_ck clock input in the receive path then allows using the phase-tuned sampling clock for the incoming data. This is required for SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

When using an external driver (a voltage switch transceiver), the SDMMC_CKIN feedback clock input can be selected to sample the receive data.

For an SD/SDIO/eMMC card, the clock frequency can vary between 0 and 208 MHz (limited by maximum I/O speed).

Depending on the selected bus mode (SDR or DDR), one bit or two bits are transferred on SDMMC_D[7:0] lines with each clock cycle. The SDMMC_CMD line transfers only one bit per clock cycle.

60.5.4 SDMMC adapter

The SDMMC adapter (see Figure 669: SDMMC block diagram) is a multimedia/secure digital memory card bus master that provides an interface to a MultiMediaCard stack or to a secure digital memory card. It consists of the following subunits:

- Control unit
- Data transmit path
- Command path
- Data receive path
- Response path
- Receive data path clock multiplexer
- Delay block (DLYB), external to the SDMMC
- Adapter register block
- Data FIFO
- Internal DMA (IDMA)

Note: The adapter registers and FIFO use the AHB clock domain (sdmmc_hclk). The control unit, command path and data transmit path use the SDMMC adapter clock domain (sdmmc_ker_ck). The response path and data receive path use the SDMMC adapter feedback clock domain from the sdmmc_io_in_ck, or SDMMC_CKIN, or from the sdmmc_fb_ck generated by DLYB.

The DLYB delay block on the device can be used in conjunction with the SDMMC adapter, to tune the phase of the sampling clock for incoming data in SDMMC receive mode. It is required for the SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

Adapter register block

The adapter register block contains all system control registers, the SDMMC command and response registers and the data FIFO.

This block also generates the signals from the corresponding bit location in the SDMMC Clear register that clear the static flags in the SDMMC adapter.
Control unit

The control unit illustrated in Figure 671, contains the power management functions, the SDMMC_CK clock management with divider, and the I/O direction management.

Figure 671. Control unit

The power management subunit disables the card bus output signals during the power-off and power-up phases.

There are three power phases:
- power-off
- power-up
- power-on

The clock management subunit uses the sdmmc_ker_ck to generate the SDMMC_CK and provides the division control. It also takes care of stopping the SDMMC_CK for i.e. flow control.

The clock outputs are inactive:
- after reset
- during the power-off or power-up phases
- if the power saving mode (register bit PWRSAV) is enabled and the card bus is in the Idle state for eight clock periods. The clock will be stopped eight cycles after both the command/response CPSM and data path DPSM subunits have entered the Idle phase. The clock will be restarted when the command/response CPSM or data path DPSM is activated (enabled).

The I/O management subunit takes care of the SDMMC_Dn and SDMMC_CMD I/O direction signals, which controls the external voltage transceivers.

Command/Response path

The Command/Response path subunit transfers commands and responses on the SDMMC_CMD line. The Command path is clocked on the SDMMC_CK and sends commands to the card. The Response path is clocked on the sdmmc_rx_ck and receives responses from the card.
When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent the CRC is appended and the command path state machine (CPSM) sets the status flags and:
- if a response is not required enters the Idle state.
- If a response is required, it waits for the response.

When the response is received,
- for a response with CRC, the received CRC code and the internally generated code are compared, and the appropriate status flag is set according the result.
- for a response without CRC, no CRC is checked, and the appropriate status flag is not set.

When ever the CPSM is active, i.e. not in the Idle state, the CPSMACT bit is set.
**Figure 673. Command path state machine (CPSM)**

- **Idle**: The command path is inactive. When the command control register is written and the enable bit (CPSMEN) is set, the CPSM will activate the SDMMC_CK clock (when stopped due to power save PWRSAV bit) and moves
  - to the Send state when WAITPEND = 0 & BOOTEN = 0.
  - to the Pending state when WAITPEND = 1.
  - to the boot state when BOOTEN = 1.
- **Send**: The command is sent and the CRC is appended.
  - When CMDTRANS bit is set or when BOOTEN bit is set and BOOTMODE is alternative boot, and the DTDIR = receive, the CPSM DataEnable signal will be issued to the DPSM at the end of the command.
  - When the CMDTRANS bit is set and the CMD SUSPEND bit is 0 the interrupt period will be terminated at the end of the command.
  - When CMDSTOP bit is set the CPSM Abort signal will be issue to the DPSM at the end of the command.
  - If no response is expected (WAITRESP = 00) the CPSM will move to the Idle state and the CMDSENT flag is set. When BOOTMODE = 1 & BOOTEN = 0 the CMDSENT flag is delayed 56 cycles after the command End bit, otherwise the
CMDSENT flag is generated immediately after the command End bit. The RESPCMDR and RESPxR registers are not modified.

- If a command response is expected (WAITRESP = not 00) the CPSM will move to the Wait state and start the response timeout.

**Wait:** The Command path waits for a response.
- When WAITINT bit is 0 the command timer starts running and the CPSM waits for a Start bit.
  a) If a Start bit is detected before the timeout the CPSM moves to the Receive state.
  b) If the timeout is reached before the CPSM detect a response start bit, the timeout flag (CTIMEOUT) is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are not modified.
- When WAITINT bit is 1, the timer is disabled and the CPSM waits for an interrupt request (Response Start bit) from one of the cards.
  a) When a Start bit is detected the CPSM moves to the Receive state.
  b) When writing WAITINT to 0 (interrupt mode abort), the host will send a response by its self and on detecting the Start bit the CPSM move to the Receive state.

**Receive:** The command response will be received. Depending the response mode bits WAITRESP in the command control register, the response can be either short or long, with CRC or without CRC. The received CRC code when present will be verified against the internally generated CRC code.
- When the CMDSUSPEND bit is set and the SDIO Response bit BS = 0 (response bit [39]), the interrupt period will be started after the response.
- When CMDTRANS bit is set and the CMDSUSPEND bit is set and the SDIO Response bit DF = 1 (response bit [32]) the interrupt period will be terminated after the response.
- When the CRC status passes or no CRC is present the CMDREND flag is set, the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
- When the CMDTRANS bit is set and the DTDIR = transmit, the CPSM DataEnable signal will be issued to the DPSM at the end of the command response.
- When the CRC status fails the CCRCFAIL flag is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.

**Pending:** According the pending WAITPEND bit in the command register, the CPSM enters the pending state.
- When DATALENGTH <= 5 bytes the CPSM moves to the Sent state and generates the DataEnable signal to start the data transfer aligned with the CMD12 Stop Transmission command.
- When DATALENGTH > 5 bytes, the CPSM DataEnable signal will be issued to the DPSM to start the data transfer. The CPSM waits for a sendCMD signal from the
DPSM before moving to the Sent state. This enables i.e. the CMD12 Stop Transmission command to be sent aligned with the data.

- When writing WAITPEND to 0, the CPSM will move to the Sent state.

- **Boot**: If the BOOTEN bit is set in the command register, the CPSM enters the boot state, and when:
  - \( \text{BOOTMODE} = 0 \) the SDMMC_CMD line is driven low and when CMDTRANS bit is set and the DTDIR = receive, the CPSM DataEnable signal will be issued to the DPSM. This enables normal boot operation. This state is left at the end of the boot procedure by clearing the register bit BOOTEN, which cause the SDMMC_CMD line to be driven high and the CPSM Abort signal will be issued to the DPSM, before moving to the Idle state. The CMDSENT flag is generated 56 cycles after SDMMC_CMD line is high.
  - \( \text{BOOTMODE} = 1 \), move to the Send state. This enables sending of the CMD0 (boot). Clearing BOOTEN has no effect.

*Note: The CPSM remains in the Idle state for at least eight SDMMC_CK periods to meet the \( N_{CC} \) and \( N_{RC} \) timing constraints. \( N_{CC} \) is the minimum delay between two host commands, and \( N_{RC} \) is the minimum delay between the host command and the card response.*

*Note: The response timeout has a fixed value of 64 SDMMC_CK clock periods.*

A Command is a token that starts an operation. Commands are sent from the host to either a single card (addressed command) or all connected cards (broadcast command are available for eMMC V3.31 or previous). Commands are transferred serially on the SDMMC_CMD line. All commands have a fixed length of 48 bits. The general format for a command token for SD-Memory cards, SDIO cards, and eMMC cards is shown in Table 438.

The Command token data is taken from 2 registers, one containing a 32-bits argument and the other containing the 6-bits command index (six bits sent to a card).

**Table 438. Command token format**

<table>
<thead>
<tr>
<th>Bit position</th>
<th>Width</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>1</td>
<td>0</td>
<td>Start bit</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>1</td>
<td>Transmission bit</td>
</tr>
<tr>
<td>[45:40]</td>
<td>6</td>
<td>x</td>
<td>Command index</td>
</tr>
<tr>
<td>[39:8]</td>
<td>32</td>
<td>x</td>
<td>Argument</td>
</tr>
<tr>
<td>[7:1]</td>
<td>7</td>
<td>x</td>
<td>CRC7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>End bit</td>
</tr>
</tbody>
</table>

Next to the Command data there are command type (WAITRESP) bits controlling the command path state machine (CPSM). These bits also determine whether the command requires a response, and whether the response is short (48 bit) or long (136 bits) long, and if a CRC is present or not.

A Response is a token that is sent from an addressed card or synchronously from all connected cards to the host as an answer to a previous received Command. All responses are sent via the command line SDMMC_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type. Response tokens R1, R2, R3, R4, R5, and R6 have various
coding schemes, depending on their content. The general formats for the response tokens for SD-Memory cards, SDIO cards, and eMMC cards are shown in Table 439, Table 440 and Table 441.

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. Most responses, except some, are protected by a CRC. Every command code word is terminated by the End bit (always 1).

The Response token data is stored in 5 registers, four containing the 32-bits card status, OCR register, argument or 127-bits CID or CSD register including internal CRC, and one register containing the 6-bits command index.

Table 439. Short response with CRC token format

<table>
<thead>
<tr>
<th>Bit position</th>
<th>Width</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>1</td>
<td>0</td>
<td>Start bit</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>0</td>
<td>Transmission bit</td>
</tr>
<tr>
<td>[45:40]</td>
<td>6</td>
<td>x</td>
<td>Command index (or reserved 111111)</td>
</tr>
<tr>
<td>[39:8]</td>
<td>32</td>
<td>x</td>
<td>Argument</td>
</tr>
<tr>
<td>[7:1]</td>
<td>7</td>
<td>x</td>
<td>CRC7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>End bit</td>
</tr>
</tbody>
</table>

Table 440. Short response without CRC token format

<table>
<thead>
<tr>
<th>Bit position</th>
<th>Width</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>1</td>
<td>0</td>
<td>Start bit</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>0</td>
<td>Transmission bit</td>
</tr>
<tr>
<td>[45:40]</td>
<td>6</td>
<td>x</td>
<td>Command index (or reserved 111111)</td>
</tr>
<tr>
<td>[39:8]</td>
<td>32</td>
<td>x</td>
<td>Argument</td>
</tr>
<tr>
<td>[7:1]</td>
<td>7</td>
<td>111111</td>
<td>(reserved 111111)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>End bit</td>
</tr>
</tbody>
</table>

Table 441. Long response with CRC token format

<table>
<thead>
<tr>
<th>Bit position</th>
<th>Width</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>135</td>
<td>1</td>
<td>0</td>
<td>Start bit</td>
</tr>
<tr>
<td>134</td>
<td>1</td>
<td>0</td>
<td>Transmission bit</td>
</tr>
<tr>
<td>[133:128]</td>
<td>6</td>
<td>111111</td>
<td>Reserved</td>
</tr>
<tr>
<td>[127:1]</td>
<td>127:8</td>
<td>x</td>
<td>CID or CSD slices</td>
</tr>
<tr>
<td></td>
<td>7:1</td>
<td>x</td>
<td>CRC7 (included in CID or CSD)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>End bit</td>
</tr>
</tbody>
</table>

The Command/Response path operates in a half-duplex mode, so that either commands can be sent or responses can be received. If the CPSM is not in the Send state, the
SDMMC_CMD output is in the Hi-Z state. Data sent on SDMMC_CMD are synchronous with the SDMMC_CK according the NEGEDGE register bit see Figure 670.

The Command and Short Response with CRC, the CRC generator calculates the CRC checksum for all 40 bits before the CRC code. This includes the start bit, transmitter bit, command index, and command argument (or card status).

For the Long Response the CRC checksum is calculated only over the 120 bits of R2 CID or CSD. Note that the start bit, transmitter bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:

\[
\text{CRC}[6:0] = \text{Remainder} \left( \left( M(x) \cdot x^7 \right) / G(x) \right)
\]

\[
G(x) = x^7 + x^3 + 1
\]

\[
M(x) = \text{(first bit)} \cdot x^n + \text{(second bit)} \cdot x^{n-1} + ... + \text{(last bit before CRC)} \cdot x^0
\]

Where \( n = 39 \) or 119.

The CPSM allows to send a number of specific commands to handle various operating modes when CPSMEN is set, see Table 442.

### Table 442. Specific Commands overview

<table>
<thead>
<tr>
<th>VSWITCH</th>
<th>BOOTEN</th>
<th>BOOTMODE</th>
<th>CMDTRANS</th>
<th>WAITPEND</th>
<th>CMDSTOP</th>
<th>WAITINT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Start Voltage Switch Sequence</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Start normal boot</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Start alternative boot</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Stop alternative boot.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Send command with associated data transfer.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>eMMC stream data transfer, command (STOP_TRANSMISSION) pending until end of data transfer.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>eMMC stream data transfer, command different from (STOP_TRANSMISSION) pending until end of data transfer.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>Send command (STOP_TRANSMISSION), stopping any ongoing data transmission.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Enter eMMC wait interrupt (Wait-IRQ) mode.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Any other none specific command</td>
</tr>
</tbody>
</table>

The Command/Response path implements the status flags and associated clear bits shown in Table 443:
Table 443. Command path status flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMDSENT</td>
<td>Set at the end of the command without response. (CPSM moves from SEND to IDLE)</td>
</tr>
<tr>
<td>CMDREND</td>
<td>Set at the end of the command response when the CRC is OK. (CPSM moves from RECEIVE to IDLE)</td>
</tr>
<tr>
<td>CCRCFAIL</td>
<td>Set at the end of the command response when the CRC is FAIL. (CPSM moves from RECEIVE to IDLE)</td>
</tr>
<tr>
<td>CTIMEOUT</td>
<td>Set after the command when no response start bit received before the timeout. (CPSM moves from WAIT to IDLE)</td>
</tr>
<tr>
<td>CKSTOP</td>
<td>Set after the voltage switch (VSWITCHEN = 1) command response when the CRC is OK and the SDMMC_CK is stopped. (no impact on CPSM)</td>
</tr>
<tr>
<td>VSWEND</td>
<td>Set after the voltage switch (VSWITCH = 1) timeout of 5ms + 1ms. (no impact on CPSM)</td>
</tr>
<tr>
<td>CPSMACT</td>
<td>Command transfer in progress. (CPSM not in Idle state)</td>
</tr>
</tbody>
</table>

The Command path error handling is shown in Table 447:

Table 444. Command path error handling

<table>
<thead>
<tr>
<th>Error</th>
<th>CPSM state</th>
<th>Cause</th>
<th>Card action</th>
<th>Host action</th>
<th>CPSM action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>Wait</td>
<td>No start bit in time</td>
<td>Unknown</td>
<td>Reset or cycle power card(^{(1)})</td>
<td>Move to Idle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRC status</td>
<td>Receive</td>
<td>Negative status</td>
<td>Command ignored</td>
<td>Resend command(^{(1)})</td>
<td>Move to Idle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transmission error</td>
<td>Command accepted</td>
<td>Resend command(^{(1)})</td>
<td></td>
</tr>
</tbody>
</table>

1. When CMDTRANS is set, also a stop_transmission command shall be send to move the DPSM to Idle.

Data path

The data path subunit transfers data on the SDMMC_D[7:0] lines to and from cards. The data transmit path is clocked on the SDMMC_CK and sends data to the card. The data receive path is clocked on the sdmmc_rx_ck and receives data from the card. Figure 674 shows the data path block diagram.
The card data bus width can be programmed in the clock control register bits WIDBUS. The supported data bus width modes are:

- If the wide bus mode is not enabled, only one bit is transferred over SDMMC_D0.
- If the 4-bit wide bus mode is enabled, data is transferred at four bits over SDMMC_D[3:0].
- If the 8-bit wide bus mode is enabled, data is transferred at eight bits over SDMMC_D[7:0].

Next to the data bus width the data sampling mode can be programmed in the clock control register bit DDR. The supported data sampling modes are:

- Single data rate signaling (SDR), data is clocked on the rising edge of the clock.
- Double data rate signaling (DDR), data is clocked on the both edges of the clock. DDR mode is only supported in wide bus mode (4-bit wide and 8-bit wide).

*Note:* The data sampling mode only applies to the SDMMC_D[7:0] lines. (not applicable to the SDMMC_CMD line.)
In DDR mode, data is sampled on both edges of the SDMMC_CK according the following rules, see also Figure 675 and Figure 676:

- On the rising edge of the clock Odd bytes are sampled.
- On the falling edge of the clock Even bytes are sampled.
- Data payload size is always a multiple of 2 Bytes.
- Two CRC16 are computed per data line
  - Odd bits CRC16 clocked on the falling edge of the clock.
  - Even bits CRC16 clocked on the rising edge of the clock.
- Start, End bits and idle conditions are full cycle.
- CRC status / boot acknowledgment and Busy signaling are full cycle and are only sampled on the rising edge of the clock.

In DDR mode the SDMMC_CK clock division shall be \( \geq 2 \).

**Figure 675. DDR mode data packet clocking**

![Data Packet Clocking Diagram](image)

**Figure 676. DDR mode CRC status / boot acknowledgment clocking**

![CRC Status/Boot Acknowledgment Diagram](image)

Data path state machine (DPSM)

Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait_S or Wait_R state when it is enabled:

- **Send**: the DPSM moves to the Wait_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- **Receive**: the DPSM moves to the Wait_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.
For boot operation with acknowledgment the DPSM moves to the Wait_Ack state and waits for the boot acknowledgment before moving to the Wait_R state.

The DPSM operates at SDMMC_CK. The DPSM has the following states, as shown in Figure 677. When ever the DPSM is active, i.e. not in the Idle state, the DPSMACT bit is set.

**Figure 677. Data path state machine (DPSM)**

- **Idle** state: the data path is inactive, and the SDMMC_D[7:0] outputs are according the PWRCTRL setting. The DPSM will be activated either by sending a command with CMDTRANS bit set or by setting the DTEN bit, or by detecting Busy on SDMMC_D0 (that is, after a command with R1b response).

  When not busy, the DPSM will activate the SDMMC_CK clock (when stopped due to power save PWRSAV bit), loads the data counter with a new (DATALENGTH) value and:
  - When the data direction bit (DTDIR) indicates send, moves to the Wait_S.
  - When the data direction bit (DTDIR) indicates receive, moves to the - Wait_R when BOOTACKEN register bit is clear.
  - Wait_Ack when BOOTACKEN register bit is set and start the acknowledgment timeout.

  When busy the DPSM will keep the SDMMC_CK clock active and move to the Busy state.
Note: *DTEN shall not be used to start data transfer with SD, SDIO and e•MMC cards.*

- **Wait_Ack** state: the data path waits for the boot acknowledgment token.
  - The DPSM moves to the Wait_R state if it receives an error free acknowledgment before a timeout.
  - When a pattern different from the acknowledgment is received an acknowledgment status error is generated, and the Ack fail status flag (ACKFAIL) is set. The DPSM stays in Wait_Ack.
  - If it reaches a timeout (ACKTIME) before it detects a start bit, it sets the timeout status flag (ACKTIMEOUT). The DPSM stays in Wait_Ack.
  - When the CPSM Abort signal is set it moves to the Idle state and sets the DABORT flag.

- **Wait_R** state: the data path, if the data counter is not zero and data is not held, waits for a start bit on SDMMC_D[n:0]. If the data counter is zero or data is hold, wait for the FIFO to be empty.
  - In block mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DBLOCKSIZE.
  - In SDIO multibyte mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DATALENGTH.
  - In stream mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data counter with DATALENGTH.
  - if the data counter (DATACOUNT) equals zero (end of data) the DPSM moves to the Idle state when the receive FIFO is empty and the DATAEND flag is set.
  - If it reaches a timeout (DATATIME) before it detects a start bit, it sets the timeout status flag (DTIMEOUT) and the DPSM stays in the WAIT_R state.
  - If the CPSM Abort signal is set:
    - If DATACOUNT > 0, the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST, and sets the DABORT flag.
    - If DATACOUNT is zero normal operation is continued, there will be no DABORT flag since the transfer has completed normally.
  - if the DTHOLD bit is set:
    - When DATACOUNT > 0, the DPSM moves to the Idle state when the receive FIFO is empty and when IDMAEN = 0 reset with FIFORST, and issues the DHOLD flag. When Holding the timeout is disabled. When an CPSM Abort signal is received during Holding, the transfer is Aborted.
- When \text{DATACOUNT} = 0, the transfer is completed normally and there will be no \text{DHOLD} flag.
  - When \text{DPSM} has been started with \text{DTEN}, after an error (\text{DTIMEOUT}) the \text{DPSM} moves to the \text{Idle} state when the \text{FIFO} is empty and when \text{IDMAEN} = 0 reset with \text{FIFORST}.

- **ReadWait** state: the data path \text{ReadWait} the bus.
  - The \text{DPSM} moves to the \text{Wait\_R} state when the \text{ReadWait} stop bit (\text{RWSTOP}) is set, and start the receive timeout.
  - If the \text{CPSM} \text{Abort} signal is set, wait for the \text{FIFO} to be empty and when \text{IDMAEN} = 0 reset with \text{FIFORST}, then moves to the \text{Idle} state and sets the \text{DABORT} flag.

- **Receive** state: the data path receives serial data from a card. Pack the data in bytes and write it to the data \text{FIFO}. Depending on the transfer mode selected in the data control register (\text{DTMODE}), the data transfer mode can be either block or stream:
  - In block mode, when the data block size (\text{DBLOCKSIZE}) number of data bytes are received, the \text{DPSM} waits until it receives the CRC code.
  - In SDIO multibyte mode, when the data block size (\text{DATALENGTH}) number of data bytes are received, the \text{DPSM} waits until it receives the CRC code.

  a) If the received CRC code matches the internally generated CRC code, the \text{DPSM} moves to the
     - \text{Wait\_R} state when \text{RWSTART} = 0 and start the receive timeout.
     - \text{ReadWait} state when \text{RWSTART} = 1 and \text{DATACOUNT} > zero, and the \text{DBCKEND} flag is set.
  
  b) If the received CRC code fails the internally generated CRC code any further data reception is prevented.
     - When not all data has been received (\text{DATACOUNT} > 0), the CRC fail status flag (\text{DCRCFAIL}) is set and the \text{DPSM} stays in the \text{Receive} state.
     - When all data has been received (\text{DATACOUNT} = 0), wait for the \text{FIFO} to be empty after which the CRC fail status flag (\text{DCRCFAIL}) is set and the \text{DPSM} moves to the \text{Idle} state.

  - In stream mode, the \text{DPSM} receives data while the data counter \text{DATACOUNT} > 0. When the counter is zero, the remaining data in the shift register is written to the data \text{FIFO}, and the \text{DPSM} moves to the \text{Wait\_R} state.

  - When a \text{FIFO} overrun error occurs, the \text{DPSM} sets the \text{FIFO} overrun error flag (\text{RXOVERR}) and any further data reception is prevented. The \text{DPSM} stays in the \text{Receive} state.

  - When a \text{CPSM\_Abort} signal is received:
     - If the \text{CPSM\_Abort} signal is received before the 2 last bits of the data with \text{DATACOUNT} = 0, the transfer is aborted. The remaining data in the shift register is written to the data \text{FIFO}, wait for the \text{FIFO} to be empty and when \text{IDMAEN} = 0 reset with \text{FIFORST}, then the \text{DPSM} moves to the \text{Idle} state and the \text{DABORT} flag is set.
     - If the \text{CPSM\_Abort} signal is received during or after the 2 last bits of the transfer with \text{DATACOUNT} = 0, the transfer is completed normally. The \text{DPSM} stays in the \text{Receive} state no \text{DABORT} flag is generated.

  - When \text{DPSM} has been started with \text{DTEN}, after an error (\text{DCRCFAIL} when \text{DATACOUNT} > 0, or \text{RXOVERR}) the \text{DPSM} moves to the \text{Idle} state when the \text{FIFO} is empty and when \text{IDMAEN} = 0 reset with \text{FIFORST}.
- **Wait_S** state: the data path waits for data to be available from the FIFO.
  - If the data counter DATACOUNT > 0, waits until the data FIFO empty flag (TXFIFOE) is deasserted and DTHOLD is not set, and moves to the Send state.
  - if the data counter (DATACOUNT) = 0 the DPSM moves to the Idle state.
    - When DTHOLD is disabled, the DATAEND flag is set.
    - When DTHOLD is enabled, the DHALD flag is set.
  - When DTHOLD is set and the DATACOUNT > 0
    - When IDMA is enabled, the DBCKEND flag is set and subsequently the FIFO is flushed, furthermore the DPSM will move to the Idle state and the DHALD flag is set.
    - When IDMA is disabled the DBCKEND flag is set. Wait for the FIFO to be reset by software with FIFORST, then DPSM will move to the Idle state and issues the DHALD flag.
  - When DTHOLD is set and DATACOUNT = 0 the transfer is completed normally.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and sets the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT=0, normal operation is continued, there will be no DABORT flag since the transfer has completed normally.

**Note:** The DPSM remains in the Wait_S state for at least two clock periods to meet the NWR timing requirements, where NWR is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

- **Send** state: the DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block, SDIO multibyte or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are send, the DPSM sends an internally generated CRC code and End bit, and moves to the Busy state and start the transmit timeout.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are send, the DPSM sends an internally generated CRC code and End bit, and moves to the Busy state and start the transmit timeout.
  - In stream mode, the DPSM sends data to a card while the data counter DATACOUNT > 0. When the data counter reaches zero moves to the Busy state and start the transmit timeout.
    - Before sending the last stream Byte according to DATACOUNT, the DPSM issues a trigger on the sendCMD signal. This signal is used by the CPSM to sent any pending command. (i.e. CMD12 Stop Transmission command)
    - If a FIFO underrun error occurs, the DPSM sets the FIFO underrun error flag (TXUNDERR). The DPSM stays in the Send state.
    - When receiving the CPSM Abort signal
      - If the CPSM Abort signal is received before the 2 last bits of the transfer with DATACOUNT=0, the transfer is aborted. The DPSM will send a last data bit followed by an End bit. The FIFO will be disabled/flushed, and the DPSM moves to the Busy state to wait for not busy before setting the DABORT flag.
      - If the CPSM Abort signal is received during or after the 2 last bits of the transfer
with DATACOUNT=0, the transfer is completed normally, there will be no DABORT flag.

- **Busy** state: the DPSM waits for the CRC status token when expected, and wait for a not busy signal:
  - If a CRC status token is expected and indicate "non-erroneous transmission" or when there is no CRC expected:
    - it moves to the Wait_S state when SDMMC_D0 is not low (the card is not busy).
    - When the card is busy SDMMC_D0 is low it will remain in the Busy state.
  - If a CRC status token is expected and indicates "erroneous transmission".
    - When not all data has been send (DATACOUNT > 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set. The FIFO will be disabled/flushed and the DPSM stays in the Busy state.
    - When all data has been send (DATACOUNT = 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - If a CRC status (Ncrc) timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - If a busy timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
    - When receiving the CPSM Abort signal in the Busy state:
      - If the CPSM Abort signal is received before the 2 last bits of the CRC response with DATACOUNT > 0, the data transfer is aborted. The DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
      - If the CPSM Abort signal is received during or after the 2 last bits of the CRC response when DATACOUNT=0 or when no CRC is expected and DATACOUNT = 0 and there has been no DTIMEOUT error, the DPSM stays in the Busy state no DABORT flag is generated, since the transfer may completed normally.
      - If the CPSM Abort signal is received when a DTIMEOUT error has occurred the DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
      - When entering the Busy state due to an Abort in the Send state, the DPSM waits for not busy before moving to the Idle state and the DABORT flag is set.
    - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or DTIMEOUT) the DPSM moves to the Idle state when the FIFO is reset.
    - When the DPSM has been started due to Busy on SDMMC_D0, waits for not busy after which the Busy end status flag (BUSYD0END) is set and the DPSM moves to the Idle state.
The data timer (DATATIME) is enabled when the DPSM is in the Wait_R or Busy state 2 cycles after the data block end bit, or data read command end bit, or R1b response, and generates the data timeout error (DTIMEOUT):

- When transmitting data, the timeout occurs
  - when a CRC status is expected and no start bit is received within 8 SDMMC_CK cycles, the DTIMEOUT flag is set.
  - when the Busy state takes longer than the programmed timeout period, the DTIMEOUT flag is set.
- When receiving data, the timeout occurs
  - when there is still data to be received DATACOUNT > 0 and no start bit is received before the programmed timeout period, the DTIMEOUT flag is set.
- After a R1b response, the timeout occurs
  - when the Busy state takes longer than the programmed timeout period, the DTIMEOUT flag is set.

When DATATIME = 0,

- In receive the start bit shall be present 2 cycles after the data block end bit or data read command end bit.
- In transmit busy is timed out 2 cycles after the CRC token end bit or stream data end bit.
- After a R1b response busy is timed out 2 cycles after the response end bit.

Data can be transferred from the card to the host (transmit, send) or vice versa (receive). Data are transferred via the SDMMC_Dn data lines, they are stored in a FIFO.

### Table 445. Data token format

<table>
<thead>
<tr>
<th>Description</th>
<th>Start bit</th>
<th>Data(1)</th>
<th>CRC16</th>
<th>End bit</th>
<th>DTMODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block data</td>
<td>0</td>
<td>(DBLOCKSIZE, DATALENGTH)</td>
<td>yes</td>
<td>1</td>
<td>00</td>
</tr>
<tr>
<td>SDIO multibyte</td>
<td>0</td>
<td>(DATALENGTH)</td>
<td>yes</td>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>eMMC stream</td>
<td>0</td>
<td>(DATALENGTH)</td>
<td>no</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

1. The total amount of data to transfer is given by DATALENGTH. Where for Block data the amount of data in each block is given by DBLOCKSIZE.

The data token format is selected with register bits DTMODE according.

The data path implements the status flags and associated clear bits shown in Table 446:

### Table 446. Data path status flags and clear bits

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATAEND</td>
<td>Set at the end of the complete data transfer when the CRC is OK and busy has finished and both DTHOLD = 0 and DATACOUNT = 0. (DPSM moves from WAIT_S to IDLE)</td>
</tr>
<tr>
<td>RX</td>
<td>Set at the end of the complete data transfer when the CRC is OK and all data has been read, (DATACOUNT = 0 and FIFO is empty). (DPSM moves from WAIT_R to IDLE)</td>
</tr>
<tr>
<td>BOOT</td>
<td>Set at the end of the complete data transfer when the CRC is OK and all data has been read, (DATACOUNT = 0 and FIFO is empty). (DPSM moves from WAIT_R to IDLE)</td>
</tr>
</tbody>
</table>
The data path error handling is shown in Table 447:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCRCFAIL</td>
<td>TX: Set at the end of the CRC when FAIL and busy has finished. (DPSM stay in BUSY when there is still data to send and wait for Abort) (DPSM moves from BUSY to IDLE when all data has been sent) or DPSM has been started with DTEN</td>
</tr>
<tr>
<td></td>
<td>RX: Set at the end of the CRC when FAIL and FIFO is empty. (DPSM stays in RECEIVE when there is still data to be received and wait for Abort) (DPSM moves from RECEIVE to IDLE when all data has been received or DPSM has been started with DTEN)</td>
</tr>
<tr>
<td>ACKFAIL</td>
<td>BOOT: Set at the end of the BOOT ACK when FAIL. (DPSM stays in Wait_Ack and wait for Abort)</td>
</tr>
<tr>
<td>DTIMEOUT</td>
<td>TX: Set when no CRC token start bit received within Ncrc, or no end of busy received before the timeout. (DPSM stays in BUSY and wait for Abort) (When DPSM has been started with DTEN move to IDLE)</td>
</tr>
<tr>
<td></td>
<td>RX: Set when no start bit received before the timeout. (DPSM stays in WAIT_R and wait for Abort) (When DPSM has been started with DTEN move to IDLE)</td>
</tr>
<tr>
<td>ACKTIMEOUT</td>
<td>BOOT: Set when no start bit received before the timeout. (DPSM stays in Wait_Ack and wait for Abort)</td>
</tr>
<tr>
<td>DBCKEND</td>
<td>TX: When DTHOLD = 1 and IDMAEN = 0: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT &gt;0). (DPSM moves from TRANSMIT to WAIT_S)</td>
</tr>
<tr>
<td></td>
<td>RX: When RWSTART = 1: Set at the end of data block transfer when the CRC is OK, when data transfer is not complete (DATACOUNT &gt; 0). (DPSM moves from RECEIVE to READWAIT)</td>
</tr>
<tr>
<td>DHOLD</td>
<td>TX: When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished. (DPSM moves from WAIT_S to IDLE)</td>
</tr>
<tr>
<td></td>
<td>RX: When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and all data has been read (FIFO is empty), when data transfer is not complete (DATACOUNT &gt;0). (DPSM moves from WAIT_R to IDLE)</td>
</tr>
<tr>
<td>DABORT</td>
<td>CMD R1b: When Abort event has been sent by the CPSM and busy has finished. (DPSM moves from BUSY to IDLE)</td>
</tr>
<tr>
<td></td>
<td>TX: When Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from Any state to IDLE)</td>
</tr>
<tr>
<td></td>
<td>RX: When Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from Any state to IDLE)</td>
</tr>
<tr>
<td></td>
<td>BOOT: Set after the command response when end of busy before the timeout. (DPSM moves from BUSY to IDLE)</td>
</tr>
<tr>
<td>BUSYD0END</td>
<td>CMD R1b: Set at the end of the CRC when FAIL and busy has finished. (DPSM moves from BUSY to IDLE)</td>
</tr>
<tr>
<td>DPSMACT</td>
<td>Data transfer in progress. (DPSM not in Idle state)</td>
</tr>
</tbody>
</table>
Table 447. Data path error handling

<table>
<thead>
<tr>
<th>Error</th>
<th>DPSM state</th>
<th>Cause</th>
<th>Card action</th>
<th>Host action</th>
<th>DPSM action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>Wait_Ack</td>
<td>No Ack in time</td>
<td>unknown</td>
<td>Card cycle power</td>
<td>Stay in Wait_Ack (reset the SDMMC with the RCC.SDMMCxRST register bit)</td>
</tr>
<tr>
<td></td>
<td>Wait_R</td>
<td>No Start bit in time</td>
<td>unknown</td>
<td>Stop data reception</td>
<td>Send stop transmission command</td>
</tr>
<tr>
<td></td>
<td>Busy</td>
<td>Busy too long (due to data transfer)</td>
<td>unknown</td>
<td>Stop data reception</td>
<td>Send stop transmission command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>On CPSM_Abort move to Idle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CRC</td>
<td>Receive transmission error</td>
<td>Send further data</td>
<td>Stop data reception</td>
<td>On CPSM_Abort move to Idle</td>
</tr>
<tr>
<td></td>
<td>CRC status</td>
<td>Busy</td>
<td>Ignore further data</td>
<td>Stop data transmission</td>
<td>Send stop transmission command</td>
</tr>
<tr>
<td></td>
<td>Ack status</td>
<td>Wait_Ack</td>
<td>transmission error</td>
<td>Stop boot procedure</td>
<td>On CPSM_Abort move to Idle</td>
</tr>
<tr>
<td></td>
<td>Overrun</td>
<td>Receive</td>
<td>FIFO full</td>
<td>Stop data reception</td>
<td>Send stop transmission command</td>
</tr>
<tr>
<td></td>
<td>Underrun</td>
<td>Send</td>
<td>FIFO empty</td>
<td>Stop data transmission</td>
<td>Send stop transmission command</td>
</tr>
</tbody>
</table>

Data FIFO

The data FIFO (first-in-first-out) subunit contains the transmit and receive data buffer. A single FIFO is used for either transmit or receive as selected by the DTDIR bit. The FIFO contains 32-bit wide, 16-word deep data buffer and control logic. Because the data FIFO operates in the AHB clock domain (sdmmc_hclk), all signals from the subunits in the SDMMC clock domain (SDMMC_CK/sdmmc_rx_ck) are resynchronized.

The FIFO can be in one of the following states:
- The transmit FIFO refers to the transmit logic and data buffer when sending data out to the card. (DTDIR = 0)
- The receive FIFO refers to the receive logic and data buffer when receiving data in from the card. (DTDIR = 1)
The end of a correctly completed SDMMC data transfer from the FIFO is indicated by the DATAEND flags driven by the data path subunit. Any incorrect (aborted) SDMMC data transfer from the FIFO is indicated by one of the error flags (DCRCFAIL, DTIMEOUT, DABORT) driven by the data path subunit, or one of the FIFO error flags (TXUNDERR, RXOVERR) driven by the FIFO control.

The data FIFO can be accessed in the following ways, see Table 448.

Table 448. Data FIFO access

<table>
<thead>
<tr>
<th>Data FIFO access</th>
<th>IDMAEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>From FW via AHB slave interface</td>
<td>0</td>
</tr>
<tr>
<td>From IDMA via AHB master interface</td>
<td>1</td>
</tr>
</tbody>
</table>

Transmit FIFO:

Data can be written to the transmit FIFO when the DPSM has been activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by FW via the AHB slave interface. The transmit FIFO is accessible via sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out. The transmit FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   - For block data transfer (DTMODE = 0), DATALENGTH shall be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in transmit mode (DTDIR = 0).
   - Configures the FIFO in transmit mode.
3. Enabled the data transfer
   - either by sending a Command from the CPSM with the CMDTRANS bit set
   - or by setting DTEN bit
4. When (DPSMACT = 1) write data to the FIFO.
   - The DPSM will stay in the Wait_S state until FIFO is full (TXFIFOF = 1), or the number indicated by DATALENGTH.
   - The SDMMC start sending data as long as FIFO is not empty.
5. When the FIFO is half empty (TXFIFOHE flag), write data to the FIFO until FIFO is full (TXFIFOF = 1), or last data has been written.
6. When last data has been written wait for end of data (DATAEND flag)
   - SDMMC has completely sent all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer error or transfer hold when IDMAEN = 0, FW shall stop writing to the FIFO and flush and reset the FIFO with the FIFORST register bit.

The transmit FIFO status flags are listed in Table 449.
Receive FIFO:
Data can be read from the receive FIFO when the DPSM is activated (DPSMACT = 1).
When IDMAEN = 1 the FIFO is fully handled by the IDMA.
When IDMAEN = 0 the FIFO is controlled by FW via the AHB slave interface. When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. The receive FIFO is accessible via sequential addresses. The receive FIFO is handled in the following way:
1. Write the data length into DATALength and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALength shall be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in receive mode (DTDIR = 1).
   – Configures the FIFO in receive mode.
3. Enable the DPSM transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit.
4. When (DPSMACT = 1) the FIFO is ready to receive data.
   – The DPSM will write the received data to the FIFO.
5. When the FIFO is half full (RXFIFOHF flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
6. When last data has been received end of data (DATAEND flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
   – SDMMC has completely received all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer hold when IDMAEN = 0, FW shall read the remaining data until the FIFO is empty and reset the FIFO with the FIFORST register bit. This will cause the DPSM to go to the Idle state (DPSMACT = 0).

In case of a data transfer error when IDMAEN = 0, FW shall stop reading the FIFO and flush and reset the FIFO with the FIFORST register bit. This will cause the DPSM to go to the Idle state (DPSMACT = 0).

The receive FIFO status flags are listed in Table 450.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXFIFO</td>
<td>Set to high when all transmit FIFO words contain valid data.</td>
</tr>
<tr>
<td>TXFIFOE</td>
<td>Set to high when the transmit FIFO does not contain valid data.</td>
</tr>
<tr>
<td>TXFIFOHE</td>
<td>Set to high when half or more transmit FIFO words are empty.</td>
</tr>
<tr>
<td>TXUNDERR</td>
<td>Set to high when an underrun error occurs. This flag is cleared by writing to the SDMMC Clear register.</td>
</tr>
</tbody>
</table>
The CLKMUX selects the source for clock sdmmc_rx_c:k to be used with the received data and command response. The receive data clock source can be selected by the clock control register bit SELCLKRX, between:

- sdmmc_io_in_c:k bus master main feedback clock.
- SDMMC_CKIN external bus feedback clock.
- sdmmc_fb_c:k bus tuned feedback clock.

The sdmmc_io_in_c:k is selected when there is no external driver, with DS and HS.

The SDMMC_CKIN is selected when there is an external driver with SDR12, SDR25, SDR50 and DDR50.

The sdmmc_fb_c:k clock input must be selected when the DLYB block on the device is used with SDR104, HS200 and optionally with SDR50 and DDR50 modes.

The sdmmc_rx_c:k source shall be changed when the CPSM and DPSM are in the Idle state.

### 60.5.5 SDMMC AHB slave interface

The AHB slave interface generates the interrupt requests, and accesses the SDMMC adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt logic.

#### SDMMC FIFO

The FIFO access is restricted to word access:

- In transmit FIFO mode
  - Data are written to the FIFO in words (32-bits) until all data according DATALENGTH has been transferred. When the DATALENGTH is not an integer
multiple of 4, the last remaining data (1, 2 or 3 bytes) are written with a word transfer.

- In receive FIFO mode
  - Data are read from the FIFO in words (32-bits) until all data according DATALENGTH has been transferred. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are read with a word transfer padded with 0 value bytes.

When accessing the FIFO with half word or byte accesses an AHB bus fault is generated.

**SDMMC interrupts**

The interrupt logic generates an interrupt request signal that is asserted when at least one of the unmasked status flags is active. A mask register is provided to allow selection of the conditions that will generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set. Some status flags require an implicit clear in the clear register.

### 60.5.6 SDMMC AHB master interface

The AHB master interface is used to transfer the data between a memory and the FIFO using the SDMMC IDMA.

**SDMMC IDMA**

Direct memory access (DMA) is used to provide high-speed transfer between the SDMMC FIFO and the memory. The AHB master optimizes the bandwidth of the system bus. The SDMMC internal DMA (IDMA) provides one channel to be used either for transmit or receive.

The IDMA is enabled by the IDMAEN bit and supports burst transfers of 8 beats.

- In transmit burst transfer mode:
  - Data are fetched in burst from memory whenever the FIFO is empty for the number of burst transfers, until all data according DATALENGTH has been transferred. When the DATALENGTH is not an integer multiple of the burst size the remaining, smaller then burst size data is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are fetched with a word transfer.

- In receive burst transfer mode:
  - Data are stored in burst in to memory whenever the FIFO contains the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst transfer the remaining, smaller then burst size data, is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are stored with halfword and or byte transfers.

In addition the IDMA provides the following channel configurations selected by bit IDMABMODE:

- single buffered channel
- double buffered channel
Single buffered channel

In single buffer configuration the data at the memory side is accessed in a linear matter starting from the base address IDMABASE0. When the IDMA has finished transferring all data the and the DPSM has completed the transfer the DATAEND flag is set.

Double buffered channel

In double buffer configuration the data at the memory side is subsequently accessed from 2 buffers, one located from base address IDMABASE0 and a second located from base address IDMABASE1. This allows firmware to process one memory buffer while the IDMA is accessing the other memory buffer. The size of the memory buffers is defined by IDMABSIZE. The buffer size shall be an integer multiple of the burst size. It is possible to update the base address of the buffers on-the-fly when the channel is enabled, the following rule apply:

- When IDMABACT bit is ‘0’ the IDMA hardware uses the IDMABASE0 to access memory. When attempting to write to this register by Firmware the write is discarded, IDMABASE0 data will not be changed. Firmware is allowed to write IDMABASE1.
- When IDMABACT bit is ‘1’ the IDMA hardware uses the IDMABASE1 to access memory. When attempting to write to this register by Firmware the write is discarded, IDMABASE1 data will not be changed. Firmware is allowed to write IDMABASE0.

When the IDMA has finished transferring the data of one buffer the buffer transfer complete flag (IDMABTC) is set and the IDMABACT bit toggles where after the IDMA continues transferring data from the other buffer. When the IDMA has finished transferring all data and the DPSM has completed the transfer the DATAEND flag is set.

The IDMABASEn address shall be word aligned.

IDMA transfer error management

An IDMA transfer error can occur:

- When reading or writing a reserved address space.

On a IDMA transfer error subsequent IDMA transfers are disabled and an IDMATE flag is set. Depending when the IDMA transfer error occurs, it will normally cause the generation of a TXUNDERR or RXOVERR error.

The behavior of the IDMATE flag depend on when the IDMA transfer error occurs during the SDMMC transfer:

- An IDMA transfer error is detected before any SDMMC transfer error (TXUNDERR, RXOVERR, DRCFAIL, or DTIMEOUT):
  - The IDMATE flag is set at the same time as the SDMMC transfer error flag.
  - The TXUNDERR, RXOVERR, DRCFAIL, or DTIMEOUT interrupt is generated.

- An IDMA transfer error is detected during a STOP_TRANSNMISSION command:
  - The IDMATE flag is set at the same time as the DABORT flag.
  - The DABORT interrupt is generated.

- An IDMA transfer error is detected at the end of the SDMMC transfer (DHOLD, or DATAEND):
  - The IDMATE flag is set at the end of the SDMMC transfer.
  - A SDMMC transfer end interrupt is generated and a DHOLD or DATAEND flag is set.
The IDMATE will be generated on an other SDMMC transfer interrupt (TXUNDErr, RXOVERr, DCRcFAIL, DTIMEOUT, DABORT, DHOLD, or DATAEND).

60.5.7 MDMA request generation

The internal trigger lines from the SDMMC allow passing direct request to MDMA controller to enable successive transfers from/to different internal RAM addresses without CPU use.

When a data transfer from/to the card completes successfully, the DATAEND flag of the status register is set. The event is signaled to an MDMA request input through the sdmmc_dataend_trg output. It can trigger the clearance of the DATAEND and CMDREND flags and, eventually, a new transfer start, through MDMA direct access to the SDMMC control and configuration registers, thus without CPU intervention.

When a command response is received successfully, the CMDREND flag of the status register is set. When a busy state following an R1b response ends, the BUSYD0 flag of the status register falls down and the BUSYD0END flag rises. The sdmmc_cmdend_trg output connected to the MDMA is set when the sequence command response associated with an eventual busy signal ends. In this way, the MDMA can manage STOP_TRANSMISSION command (needed to support open mode transfers) by clearing CMDREND and BUSYD0END status flags.

When using LINUX operating system, data to be transferred through SDMMC bus are contained in separate 1-4 Kbyte blocks of the device internal memory at non-consecutive addresses. The double buffer mode allows changing the address targeted by the IDMA in the internal memory. Each time a buffer transfer is completed, the IDMABTC flag of the status register is set. By signaling this event to MDMA through the sdmmc_buffend_trg output connected to an MDMA request input, the new buffer address base can alternatively fill the IDMABASE0 / IDMABASE1 fields without CPU intervention.

The actions to program in the MDMA according to the SDMMC requests are provided in the following table:

<table>
<thead>
<tr>
<th>Trigger signal</th>
<th>Event signaled</th>
<th>Event occurrence condition</th>
<th>MDMA transfer configuration</th>
<th>MDMA action</th>
</tr>
</thead>
<tbody>
<tr>
<td>sdmmc_dataend_trg</td>
<td>End of successful data transfer</td>
<td>DATAEND = 1</td>
<td>single</td>
<td>Set DATAENDC</td>
</tr>
<tr>
<td>sdmmc_cmdend_trg</td>
<td>End of command sequence</td>
<td>CMDSENt = 1, or (CMDREnd = 1 and BUSYD0 = 0)</td>
<td>single</td>
<td>Set CMDSENtC, Set CMDREndC, Set BUSYD0ENDC</td>
</tr>
<tr>
<td>sdmmc_buffend_trg</td>
<td>End of buffer reached</td>
<td>IDMABTC = 1</td>
<td>link list</td>
<td>Set IDMABTC, Update IDMABASE0/1</td>
</tr>
</tbody>
</table>

60.5.8 AHB and SDMMC_CK clock relation

The AHB shall at least have 3x more bandwidth than the SDMMC bus bandwidth i.e. for SDR50 4-bit mode (50Mbyte/s) the minimum sdmmc_hclk frequency is 37.5MHz (150Mbyte/s).
60.5.9 Hardware flow control

The hardware flow control functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

The behavior is to stop SDMMC_CK during data transfer and freeze the SDMMC state machines. The data transfer is stalled when the FIFO is unable to transmit or receive data. The data transfer remains stalled until the transmit FIFO is half full or all data according DATALENGTH has been stored, or until the receive FIFO is half empty. Only state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled or emptied even if flow control is activated.

To enable hardware flow control, the HWFC_EN register bit must be set to 1. After reset hardware flow control is disabled.

Hardware flow control shall only be used when the SDMMC_Dn data is cycle-aligned with the SDMMC_CK. Whenever the sdmmc_fb_ck from the DLYB delay block is used, i.e in the case of SDR104 mode with a tOP and DtOP delay > 1 cycle, hardware flow control can NOT be used.

![Figure 679. Hardware flow timing](MSv40949V1)
60.6 Card functional description

60.6.1 SD I/O mode

The following features are SDMMC specific operations:

- SDIO interrupts
- SDIO suspend/resume operation (write and read suspend)
- SDIO read wait operation by stopping the clock
- SDIO read wait operation by SDMMC_D2 signaling

### Table 453. SDIO special operation control

<table>
<thead>
<tr>
<th>Operation mode</th>
<th>SDIOEN</th>
<th>RWMOD</th>
<th>RWSTOP</th>
<th>RWSTART</th>
<th>DTDIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt detection</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Suspend/Resume operation</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ReadWait SDMMC_CK clock stop (START)</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ReadWait SDMMC_CK clock stop (STOP)</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ReadWait SDMMC_D2 signaling (START)</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ReadWait SDMMC_D2 signaling (STOP)</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**SD I/O interrupts**

To allow the SD I/O card to interrupt the host, an interrupt function is available on pin 8 (shared with SDMMC_D1 in 4-bit mode) on the SD interface. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the host or deasserted due to the end of the interrupt period. After the host has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDMMC_D[3:0]).

In SD 1-bit mode pin 8 is dedicated to the interrupt function (IRQ), and there are no timing constraints on interrupts.

In SD 4-bit mode the host samples the level of pin 8 (SDMMC_D1/IRQ) into the interrupt detector only during the interrupt period. At all other times, the host interrupt ignores this value. The interrupt period begins when interrupts are enabled at the card and SDIOEN bit is set see register settings in Table 453.

In 4-bit mode the card can generate a synchronous or asynchronous interrupt as indicated by the card CCCR register SAI and EAI bits.

- Synchronous interrupt, require the SDMMC_CK to be active.
- Asynchronous interrupt, can be generated when the SDMMC_CK is stopped, 4 cycles after the start of the card interrupt period following the last data block.
The timing of the interrupt period is depended on the bus speed mode:

In DS, HS, SDR12, and SDR25 mode, selected by register bit BUSSPEED, the interrupt period is synchronous to the SD clock.

- The interrupt period ends at the next clock from the End bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set.
- The interrupt period resumes 2 SDMMC_CK after the completion of the data block.
- At the data block gap the interrupt period is limited to 2 SDMMC_CK cycles.

**Note:** DTEN shall not be used to start data transfer with SD and eMMC cards.
In SDR50, SDR104, and DDR50, selected by register bit BUSSPEED, due to propagation delay from the card to host, the interrupt period is asynchronous.

- The card interrupt period ends after 0 to 2 SDMMC_CK cycles after the End bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set. At the host the interrupt period ends after the End bit of a command that transfers data block(s). A card interrupt issued in the 1 to 2 cycles after the command End bit are not detected by the host during this interrupt period.
- The card interrupt period resumes 2 to 4 SDMMC_CK after the completion of the last data block. The host will resume the interrupt period always 2 cycles after the last data block.
- There is NO interrupt period at the data block gap.

Note: DTEN shall not be used to start data transfer with SD and eMMC cards.
When transferring Open-ended multiple block data and using DTMODE “block data transfer ending with STOP_TRANSMISSION command”, the SDMMC will mask the interrupt period after the last data block until the end of the CMD12 STOP_TRANSMISSION command.

The interrupt period is applicable for both memory and I/O operations.

In 4-bit mode, interrupts can be differentiated from other signaling according to Table 454.
SD I/O suspend and resume

This function is NOT supported in SDIO version 4.00 or later.

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the eMMC/SD bus. To share access to the host among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the host can temporarily halt (suspend) a data transfer operation to one function or memory to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is restarted (resume) where it left off.

To perform the suspend/resume operation on the bus, the host performs the following steps:

1. Determines the function currently using the SDMMC_D[3:0] line(s)
2. Requests the lower-priority or slower transaction to suspend
3. Waits for the transaction suspension to complete
4. Begins the higher-priority transaction
5. Waits for the completion of the higher priority transaction
6. Restores the suspended transaction

The card receiving a suspend command will respond with its current bus status. Only when the bus has been suspended by the card the bus status will indicated suspension completed.

There are different suspend cases conditions:

- Suspend request accepted prior to the start of data transfer.
- Suspend request not accepted, (due to data being transfered at the same time), the host keeps checking the request until it is accepted. (data transfer has suspended)
- Suspend request during write busy.
- Suspend request with write multiple.
- Suspend request during ReadWait.

For the host to know if the bus has been released it shall check the status of the suspend request, suspension completed.

When the bus status of the suspend request response indicates suspension completed, the card has released the bus. At this time the state of the suspended operation shall be saved where after an other operation can start.

The suspend command shall be sent with the CMDSUSPEND bit set. This allows to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).
The hardware does not save the number of remaining data to be transferred when resuming the suspended operation. It is up to firmware to determine the data that has been transferred and resume with the correct remaining number of data bytes.

While receiving data from the card, the SDMMC can suspend the read operation after the read data block end (DPSM in Wait_R). After receiving the suspend acknowledgment response from the card the following steps shall be taken by firmware:

1. The normal receive process shall be stopped by setting DTHOLD bit.
   a) The remaining number of data bytes in the FIFO shall be read until the receive FIFO is empty (RXFIFOE flag is set), and when IDMAEN = 0 the FIFO shall be reset with FIFORST.

2. The confirmation that all data has been read from the FIFO, and that the suspend is completed is indicated by the DHOLD flag.
   a) The remaining number of data bytes (multiple of data blocks) still to be read when resuming the operation shall be determined from the remaining number of bytes indicated by the DATACOUNT.

**Note:** When a DTIMEOUT flag occurs during the suspend procedure, this shall be ignored.

To resume receiving data from the card, the following steps shall be taken by firmware:

1. The remaining number of data bytes (multiple of data blocks) shall be programmed in DATALENGTH.
2. The DPSM shall be configured to receive data in the DTDIR bit.
3. The resume command shall be sent from the CPSM, with the CMDTRANS bit set and the CMDSUSPEND bit set, which will end the interrupt period when data transfer is resumed (response bit DF = 1) and enabled the DPSM, after which the card will resume sending data.

While sending data to the card, the SDMMC can suspend the write operation after the write data block CRC status end (DPSM in Busy). Before sending the suspend command to the card the following steps shall be taken by firmware:

1. Enable DHOLD flag (and DBCKEND flag when IDMAEN = 0)
2. The DPSM shall be prevented from start sending a new data block by setting DTHOLD.
3. When IDMAEN = 0: When receiving the DBCKEND flag the data transfer is stopped. Firmware can stop filling the FIFO, after which the FIFO shall be reset with FIFORST. Any bytes still in the FIFO need to be rewritten when resuming the operation.
4. When receiving the DHOLD flag the data transfer is stopped. The remaining number of data bytes still to be written when resuming shall be determined from the remaining number of bytes indicated by the DATACOUNT.
5. To suspend the card the suspend command shall be sent by the CPSM with the CMDSUSPEND bit set. This allows to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

To resume sending data to the card, the following steps shall be taken by firmware:

1. The remaining number of data bytes shall be programmed in DATALENGTH.
2. The DPSM shall be configured for transmission with DTDIR set and enabled by having the CPSM send the resume command with the CMDTRANS bit set and the CMDSUSPEND bit set. This will end the interrupt period and start the data transfer.
The DPSM will either go to the Wait_S state when SDMMC_D0 does not signal busy, or will go to the Busy state when busy is signaled.

3. When IDMAEN = 1: The IDMA needs to be reprogrammed for the remaining bytes to be transferred.

4. When IDMAEN = 0: Firmware shall start filling the FIFO with the remaining data.

SD I/O ReadWait

There are 2 methods to pause the data transfer during the Block gap:

1. Stopping the SDMMC_CK.
2. Using ReadWait signaling on SDMMC_D2.

The SDMMC can perform a ReadWait with register settings according Table 453.

Depending the SDMMC operation mode (DS, HS, SDR12, SDR25) or (SDR50, SDR104, DDR) each method has a different characteristic.

The timing for pause read operation by stopping the SDMMC_CK for DS, HS, SDR12, and SDR25, the SDMMC_CK may be stopped 2 SDMMC_CK cycles after the End bit. When ready the host resumes by restarting clock, see Figure 685.

**Figure 685. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25**

![Figure 685](msv40193v2.png)

The timing for pause read operation by stopping the SDMMC_CK for SDR50, SDR104, and DDR50, the SDMMC_CK may be stopped minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, after the End bit. When ready the host resumes by restarting clock, see Figure 686. (In DDR50 mode the SDMMC_CK shall only be stopped after the falling edge, when the clock line is low.)

**Figure 686. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104**

![Figure 686](msv40194v2.png)
In ReadWait SDMMC_CK clock stopping, when RWSTART is set, the DSPM stops the clock after the End bit of the current received data block CRC. The clock start again after writing 1 to the RWSTOP bit, where after the DPSM waits for a Start bit from the card.

As SDMMC_CK is stopped, no command can be issued to the card. During a ReadWait interval, the SDMMC can still detect SDIO interrupts on SDMMC_D1.

The optional ReadWait signaling on SDMMC_D2 (RW) operation is defined only for the SD 1-bit and 4-bit modes. The ReadWait operation allows the host to signal a card that is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O device. To determine when a card supports the ReadWait protocol, the host must test capability bits in the internal card registers.

The timing for ReadWait with a SDMMC_CK less then 50MHz (DS, HS, SDR12, SDR25) is based on the interrupt period generated by the card on SDMMC_D1. The host by asserting SDMMC_D2 low during the interrupt period requests the card to enter ReadWait. To exit ReadWait the host shall raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z, see Figure 687.

**Figure 687. ReadWait with SDMMC_CK < 50 MHz**

For SDR50, SDR104 with a SDMMC_CK more than 50MHz, and DDR50, the card will treat the ReadWait request on SDMMC_D2 as an asynchronous event. The host by asserting SDMMC_D2 low after minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, request the card to enter ReadWait. To exit ReadWait the host shall raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z. The host shall raise SDMMC_D2 on the SDMMC_CK clock (see Figure 688).
In ReadWait SDMMC_D2 signaling, when RWSTART is set, the DPSM drives SDMMC_D2 after the End bit of the current received data block CRC. The ReadWait signaling on SDMMC_D2 will be removed when writing 1 to the RWSTOP bit. The DPSM remains in ReadWait state for two more SDMMC_CK clock cycles to drive SDMMC_D2 to 1 for one clock cycle (in accordance with SDIO specification), where after the DPSM waits for a Start bit from the card.

During the ReadWait signaling on SDMMC_D2 commands can be issued to the card. During the ReadWait interval, the SDMMC can detect SDIO interrupts on SDMMC_D1.

### 60.6.2 CMD12 send timing

CMD12 is used to stop/abort the data transfer, the card data transmission is terminated two clock cycles after the End bit of the Stop Transmission command.

<table>
<thead>
<tr>
<th>Data operation</th>
<th>Stop Transmission command CMD12 Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC stream write</td>
<td>The data transfer is stopped/aborted by sending the Stop Transmission command.</td>
</tr>
<tr>
<td>SDMMC open ended multiple block write</td>
<td>The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.</td>
</tr>
<tr>
<td>SDMMC block write with predefined block count</td>
<td>The Stop Transmission command is not required at the end of this type of multiple block write. (sending the Stop Transmission command after the card has received the last block is regarded as an illegal command.) If the card detects an error, the host must abort the operation by sending the Stop Transmission command.</td>
</tr>
<tr>
<td>SDMMC stream read</td>
<td>The data transfer is stopped/aborted by sending the Stop Transmission command.</td>
</tr>
</tbody>
</table>
All data write and read commands can be aborted any time by a Stop Transmission command CMD12. The following data abort procedure applies during an ongoing data transfer:

1. Load CMD12 Stop Transmission command in registers and set the CMDSTOP bit.
   a) This causes the CPSM to generate the Abort signal when the command is sent to the DPSM.

2. Configure the CPSM to send a command immediately (clear WAITPEND bit).
   a) The card, when sending data, will stop data transfer 2 cycles after the Stop Transmission command End bit.
      The card when no data is being sent, will not start sending any new data.
   b) The host, when sending data, will send one last data bit followed by an End bit after the Stop Transmission command End bit.
      The host when not sending data, will not start sending any new data.

3. When IDMAEN = 0, the FIFO need to be reset with FIFORST.
   a) When writing data to the card. On the CMDREND flag FW shall stop writing data to the FIFO. Subsequently the FIFO shall be reset with FIFORST, this will flush the FIFO.
   b) When reading data from the card. On the CMDREND flag FW shall read the remaining data from the FIFO. Subsequently the FIFO shall be reset with FIFORST.

4. When IDMAEN = 1, hardware will take care of the FIFO.
   a) When writing data to the card. On the Abort signal hardware will stop the IDMA and subsequently the FIFO will be flushed.
   b) When reading data from the card. On the Abort signal hardware will instruct the IDMA to transfer the remaining data from the FIFO to RAM.

5. When the FIFO is empty/reset the DABORT flag will be generated.

**Stream operation and CMD12**

To stop the stream transfer after the last byte to be transferred, the CMD12 End bit timing shall be sent aligned with the data stream end of last byte. The following write stream data procedure applies:
1. Initialize the stream data in the DPSM, DTMODE = MCC stream data transfer.
2. Send the WRITE_DATA_STREAM command from the CPSM with CMDTRANS = 1.
3. Preload CMD12 in command registers, with the CMDSTOP bit set.
4. Configure the CPSM to send a command only after a wait pending (WAITPEND = 1) end of last data (according DATALENGTH).
5. Enabling the CPSM to send the STOP_TRANSMISSION command, the stream data End bit and command End bit will be aligned.
   a) When DATALENGTH > 5 bytes, Command CMD12 will be waited in the CPSM to be aligned with the data transfer End bit.
   b) When DATALENGTH < 5 bytes, Command CMD12 will be started before and the DPSM will remain in the Wait_S state to align the data transfer end with the CMD12 End bit.
6. The write stream data can be aborted any time by clearing the WAITPEND bit. This will cause the Preloaded CMD12 to be sent immediately and stop the write data stream.

Figure 689. CMD12 stream timing

To stop the read stream transfer after the last byte, the CMD12 End bit timing shall occur after the last data stream byte. The following read stream data procedure applies:
1. Wait for all data to be received by the DPSM (DATAEND flag).
   a) The DPSM will not receive more data than indicated by DATALENGTH, even if the card is sending more data.
2. Send CMD12 by the CPSM.
   a) CMD12 will stop the card sending data.

Note: The SDMMC will not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

Block operation and CMD12

To stop block transfer at the end of the data, the CMD12 End bit shall be sent after the last block End bit.

When writing data to the card the CMD12 End bit shall be sent after the write data block CRC token End bit. This requires the CMD12 sending to be tied to the data block transmission timing. To stop an Open-ended Multiple block write, the following procedure applies:
1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be sent by the DPSM and the CRC token to be received, (DATAEND flag).
   a) The DPSM will not send more data than indicated by DATALENGTH.
3. Send CMD12 by the CPSM.
   a) CMD12 will set the card to Idle mode.

When reading data from the card the CMD12 End bit shall be sent earliest at the same time as the card read data block last data bit. This requires the CMD12 sending to be tied to the data block reception timing. The following stop Open-ended Multiple block read data block procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be received by the DPSM (DATAEND flag).
   a) The DPSM will not receive more data than indicated by DATALENGTH, even if the card is sending more data.
3. Send CMD12 with CMDSTOP bit set by the CPSM.
   a) CMD12 will stop the Card sending more data and set the card to Idle mode. Any ongoing block transfer will be aborted by the Card.

*Note:* The SDMMC will not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

### 60.6.3 Sleep (CMD5)

The eMMC card may be switched between a Sleep state and a Standby state by CMD5. In the Sleep state the power consumption of the card is minimized and the Vcc power supply may be switched off.

The CMD5 (SLEEP) is used to initiate the state transition from Standby state to Sleep state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Sleep state is reached when the card stops pulling down the SDMMC_D0 line.

To set the card into Sleep state the following procedure applies:
1. Enable interrupt on BUSYD0END.
2. Send CMD5 (SLEEP).
3. On BUSYD0END interrupt, card is in Sleep state
4. Vcc power supply is allowed to be switched off

The CMD5 (AWAKE) is used to initiate the state transition from Sleep state to Standby state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Standby state is reached when the card stops pulling down the SDMMC_D0 line.

To set the card into Sleep state the following procedure applies:
1. Switch on Vcc power supply and wait unit minimum operating level is reached.
2. Enable interrupt on BUSYD0END.
3. Send CMD5 (AWAKE).
4. On BUSYD0END interrupt card is in Standby state.

The Vcc power supply is allowed to be switched off only after the Sleep state has been reached. The Vcc supply shall be reinstalled before CMD5 (AWAKE) is sent.
60.6.4 Interrupt mode (Wait-IRQ)

The host and card enter and exit interrupt mode (Wait-IRQ) simultaneously. In interrupt mode there is no data transfer. The only message allowed is an interrupt service request response from the card or the host. For the interrupt mode to work correctly the SDMMC_CK frequency shall be set in accordance with the achievable SDMMC_CMD data rate in Open Drain mode, which depend on the capacitive load and pull-up resistor. The CLKDIV shall be set >1, and the SETCLKRX shall select either the sdmmc_io_in_ck or SDMMC_CLKin source.

The host must ensure that the card is in Standby state before issuing the CMD40 (GO_IRQ_STATE). While waiting for an interrupt response the SDMMC_CK clock signal must be kept active.

A card in interrupt mode (IRQ state):
- is waiting for an internal card interrupt event. Once the event occurs, the card starts to send the interrupt service request response. The response is sent in open-drain mode.
- while waiting for the internal card interrupt event, the card also monitors the SDMMC_CMD line for a Start bit. Upon detection of a Start bit the card will abort the interrupt mode and switch to Standby state.

The host in interrupt mode (CPSM Wait state waiting for interrupt):
- is waiting for a card interrupt service request response (Start bit).
- while waiting for a card interrupt service request response the host may abort the interrupt mode (by clearing the WAITINT register bit), which causes the host to send a interrupt service request response R5 with RCA = 0x0000 in open-drain mode.

When sending the interrupt service request response, the sender bit-wise monitors the SDMMC_CMD bit stream. The sender whose interrupt service request response bit does not correspond to the bit on the SDMMC_CMD line stops sending. In the case of multiple senders only one will successfully send its full interrupt service request response. i.e. If the host sends simultaneously, it will lose sending after the transmission bit.

To handle the interrupt mode, the following procedure applies:
1. Set the SDMMC_CK frequency in accordance with the achievable SDMMC_CMD data rate in Open-drain mode, CLKDIV shall be set >1, and SETCLKRX shall select the sdmmc_io_in_ck.

2. Load CMD40 (GO_IRQ_STATE) in the command registers.

3. Enable wait for interrupt by setting WAITINT register bit.

4. Configure the CPSM to send a command immediately.
   a) This will cause the CMD40 to be sent and the CPSM to be halted in the Wait state, waiting for an interrupt service request response.

5. To exit the wait for interrupt state (CPSM Wait state):
   a) Upon the detection of an interrupt service request response Start bit the CPSM moves to the Receive state where the response is received. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.
   b) To abort the interrupt mode the host clears the WAITINT register bit, which will cause the host to send an interrupt service request response by itself. Which will move the CPSM to the Receive state. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.

Note: On a simultaneous send interrupt service request response Start bit collision the host will lose the bus access after the Transmission bit.

60.6.5 Boot operation

In boot operation mode the host can read boot data from the card by either one of the 2 boot operation functions:
1. Normal boot. (keeping CMD line low)
2. Alternative boot (sending CMD0 with argument 0xFFFFFFFFA)

The boot data can be read according the following configuration options, depending on card register settings:
- The partition from which boot data is read (EXT_CSD Byte[179])
- The boot data size (EXT_CSD Byte[226])
- The bus configuration during boot (EXT_CSD Byte[177])
- Receiving boot acknowledgment from the card. (EXT_CSD Byte[179])

If boot acknowledgment is enabled the card send pattern 010 on SDMMC_D0 within 50ms after boot mode has been requested by either CMD line going low or after CMD0 with argument 0xFFFFFFFFA. A boot acknowledgment timeout (ACKTIMEOUT) and acknowledgment status (ACKFAIL) is provided.

Normal boot operation

If the SDMMC_CMD line is held low for at least 74 clock cycles after card power-up or reset, before the first command is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD line goes low, the card starts to sent the first boot code data on the SDMMC_Dn line(s). The host must keep the SDMMC_CMD line low until after all boot data has been read. The host can terminate boot mode by pulling the SDMMC_CMD line high.
To perform the normal boot procedure the following steps needed:

1. Reset the card.

2. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKFAIL and ACKTIMEOUT interrupt.

3. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data bytes to be received in DATALENGTH.

4. Enable the DTIMEOUT, DATAEND, and CMDSENT interrupts for end of boot command confirmation.

5. Select the normal boot operation mode in BOOMODE, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This will cause:
   - the SDMMC_CMD to be driven low. (BOOTMODE = normal boot).
   - the ACK timeout to start.
   - DPSM to be enabled.

6. The incorrect reception of the boot acknowledgment can be detected with ACKFAIL flag or ACKTIMEOUT flag when enabled.
   - when an incorrect boot acknowledgment is received the ACKFAIL flag occurs.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.

7. when all boot data has been received the DATAEND flag will occur.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.

8. When last data has been received, read data from the FIFO until FIFO is empty (RXFIFOE = 1) after which end of data DATAEND flag is generated.
   - SDMMC has completely received all data and the DPSM is disabled.

9. The boot procedure will be terminated by FW clearing BOOTEN, which will cause the SDMMC_CMD line to go high. The CMDSENT flag is generated 56 cycles later to indicate that a new command can be sent.
   a) If the boot procedure is aborted by FW before all data has been received the CPSM Abort signal will stop data reception and disable the DPSM which will trigger an DABORT flag when enabled.

10. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command.
Alternative boot operation

After card power-up or reset, if the host send CMD0 with the argument 0xFFFFFFFA after 74 clock cycles before CMD0 is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD0 with argument 0xFFFFFFFA has been sent, the card starts to send the first boot code data on the SDMMC_Dn line(s). The master terminates boot operation by sending CMD0 (Reset).

Figure 692. Alternative boot mode operation

To perform the alternative boot procedure the following steps needed:

1. Move the SDMMC to power-off state, and reset the card
2. Move the SDMMC to power-on state. This will guarantee the 74 SCDMMC_CK cycles to be clocked before any command.
3. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKTIMEOUT flag.
4. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data to be received in DATALENGTH. Enable the DTIMEOUT and DATAEND flags.
5. Select the alternative boot operation mode in BOOTMODE, load the CMD0 with the 0xFFFFFFFA argument in the command registers. Enable CMDSENT flag for end of
Secure digital input/output MultiMediaCard interface (SDMMC) RM0455

boot command confirmation, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This will cause:
- the loaded command and argument to be sent out. (BOOTMODE = alternative boot).
- the ACK timeout to start.
- DPSM to be enabled.

6. When the command has been sent the CMDSENT flag is generated, at which time the BOOTEN bit shall be cleared.

7. The reception of the boot acknowledgment can be detected with ACKFAIL flag when enabled.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag will occur.

8. When all boot data has been received the DATAEND flag will occur.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.

9. When last data has been received, read data from the FIFO until FIFO is empty (RXFIFOE = 1) after which end of data DATAEND flag is generated.
   - SDMMC has completely received all data and the DPSM is disabled.

10. The BOOTEN bit shall be cleared, before terminating the boot procedure by sending CMD0 (Reset) with BOOTMODE = alternative boot. This will cause the CMDSENT flag to occur 56 cycles after the Command.
    - if the boot procedure is aborted by FW before all data has been received the CPSM Abort signal will stop the data transfer and disable the DPSM which will trigger an DABORT flag when enabled.

11. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command. When the RESET command has been sent successfully, the BOOTMODE control bit has to be cleared to terminate the boot operation.

60.6.6 Response R1b handling

When sending commands which have a R1b response the busy signaling is reflected in the BUSYD0 register bit and the release of busy with the BUSYD0END flag. The SDMMC_D0 line is sampled at the end of the R1b response and signaled in the BUSYD0 register bit. The BUSYD0 register bit is reset to not busy when the SDMMC_D0 line release busy, at the same time the BUSYD0END flag is generated.

Figure 693. Command response R1b busy signaling
The expected maximum busy time shall be set in the DATETIME register before sending the command. When enabled, the DTIMEOUT flag will be set when after the R1b response busy stays active longer then the programmed time.

To detect the SDMMC_D0 busy signaling when sending a Command with R1b response the following procedure applies:

- Enable CMDREND flag
- Send Command through CPSM.
- On the CMDREND flag check the BUSYD0 register bit.
  - If BUSYD0 signals not busy, signal busy release to the Firmware
  - If BUSYD0 signals busy, wait for BUSYD0END flag
- On BUSYD0END flag signal busy released to the firmware.
- On DTIMEOUT flag busy is active longer then programmed time.

### 60.6.7 Reset and card cycle power

#### Reset

Following reset the SDMMC will be in the reset state. In this state the SDMMC is disabled and no command nor data can be transferred. The SDMMC_D[7:0], and SDMMC_CMD are in HiZ and the SDMMC_CK is driven low.

Before moving to the power-on state the SDMMC shall be configured.

In the power-on state the SDMMC_CK clock is running. First 74 SDMMC_CK cycles will be clocked after which the SDMMC is enabled and command and data can be transferred.

The SDMMC states are controlled by Firmware with the PWRCTL register bits according to Figure 694.

#### Card cycle power

To perform a card cycle power the following procedure applies:
1. Reset the SDMMC with the RCC.SDMMCxRST register bit. This will reset the SDMMC to the reset state and the CPSM and DPSM to the Idle state.
2. Disable the Vcc power to the card.
3. Set the SDMMC in power-cycle state. This will make that the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low, to prevent the card from being supplied through the signal lines.
4. After minimum 1ms enable the Vcc power to the card.
5. After the power ramp period set the SDMMC to the power-off state for minimum 1ms. The SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are set to drive “1”.
6. After the 1ms delay set the SDMMC to power-on state in which the SDMMC_CK clock will be enabled.
7. After 74 SDMMC_CK cycles the first command can be sent to the card.

**Figure 695. Card cycle power / power up diagram**

<table>
<thead>
<tr>
<th>SDMMC state</th>
<th>Reset</th>
<th>Power-cycle</th>
<th>Power-off</th>
<th>Power-on</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC_CMD</td>
<td>HiZ</td>
<td>Driven '0'</td>
<td>HiZ</td>
<td>HiZ</td>
</tr>
<tr>
<td>SDMMC_Dn</td>
<td>HiZ</td>
<td>Driven '0'</td>
<td>Driven '0'</td>
<td>Driven '0'</td>
</tr>
<tr>
<td>Card Vcc</td>
<td></td>
<td></td>
<td></td>
<td>Power stable</td>
</tr>
<tr>
<td>Vcc min</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDMMC_CK</td>
<td></td>
<td>Driven '0'</td>
<td></td>
<td>Driven '1'</td>
</tr>
<tr>
<td>CMD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

60.7 **Ultra-high-speed phase I (UHS-I) voltage switch**

UHS-I mode (SDR12, SDR25, SDR50, SDR104, and DDR50) requires the support for 1.8V signaling. After power up the card starts in 3.3V mode. CMD11 invokes the voltage switch sequence to the 1.8V mode. When the voltage sequence is completed successfully the card enters UHS-I mode with default SDR12 and card input and output timings are changed.
To perform the signal voltage switch sequence the following steps are needed:

1. Before starting the Voltage Switch procedure, the SDMMC_CK frequency shall be set in the range 100 kHz - 400 kHz.

2. The host starts the Voltage Switch procedure by setting the VSWITCHEN bit before sending the CMD11.

3. The card returns an R1 response.
   - if the response CRC is pass, the Voltage Switch procedure continues the host will no longer drive the CMD and SDMMC_D[3:0] signals until completion of the voltage switch sequence. Some cycles after the response the SDMMC_CK will be stopped and the CKSTOP flag will be set.
   - if the response CRC is fail (CCRCFAIL flag) or no response is received before the timeout (CTIMEOUT flag), the Voltage Switch procedure is stopped.

4. The card drives CMD and SDMMC_D[3:0] to low at the next clock after the R1 response.

5. The host, after having received the R1 response, may monitor the SDMMC_D0 line using the BUSYD0 register bit. The SDMMC_D0 line is sampled two SDMMC_CK clock cycles after the Response. The Firmware may read the BUSYD0 register bit following the CKSTOP flag.
   - When the BUSYD0 is detected low the host FW will switch the Voltage regulator to 1.8V, after which it instructs the SDMMC to start the timing critical section of the
Voltage Switch sequence by setting register bit VSWITCH. The hardware will continue to stop the SDMMC_CK by holding it low for at least 5ms.

- When the BUSYD0 is detected high the host will abort the Voltage Switch sequence and cycle power the card.

6. The card after detecting SDMMC_CK low will begin switching signaling voltage to 1.8V.

7. The host SDMMC hardware after at least 5ms will restart the SDMMC_CK.

8. The card within 1ms from detecting SDMMC_CK transition will drive CMD and DAT[3:0] high for at least 1 SDMMC_CK cycle and then stop driving CMD and DAT[3:0].

9. The host SDMMC hardware, 1ms after the SDMMC_CK has been restarted, the SDMMC_D0 is sampled into BUSYD0 and the VSWEND flag is set.

10. The host, on the VSWEND flag, will check SDMMC_D0 line using the BUSYD0 register bit, to confirm completion of voltage switch sequence:

- When BUSYD0 is detected high, Voltage Switch has been completed successfully.
- When BUSYD0 is detected low, Voltage Switch has failed, the host will cycle power the card power.

The minimum 5ms time to stop the SDMMC_CK is derived from the internal ungated SDMMC_CK clock, which will have a maximum frequency of 25MHz (SD mode), as set by the clock divider CLKDIV. The >5ms time will be counted by $2^{12}$ cycles (10.24ms @ 400 kHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time for the SDMMC_CK clock to be stopped will be longer.

The maximum 1 ms time for the card to drive the SDMMC_Dn and SDMMC_CMD lines high is derived from the internal ungated SDMMC_CK which will have a maximum frequency of 25MHz (SD mode), as set by the clock divider CLKDIV. The SDMMC will check the lines after >1ms time which will be counted by $2^{9}$ cycles (1.28ms @25MHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time to check the lines will be longer.

The signal voltage level is supported through an external voltage translation transceiver i.e. ST6G3244ME.
To interface with an external driver (a voltage switch transceiver), next to the standard signals the SDMMC uses the following signals:

**SDMMC_CKIN** feedback input clock

**SDMMC_CDIR** I/O direction control for the CMD signal.

**SDMMC_D0DIR** I/O direction control for the SDMMC_D0 signal.

**SDMMC_D123DIR** I/O direction control for the SDMMC_D1, SDMMC_D2 and SDMMC_D3 signals.

The voltage transceiver signals **EN** and **SEL** are to be handled through general-purpose I/O.

The polarity of the SDMMC_CDIR, SDMMC_D0DIR and SDMMC_D123DIR signals can be selected through SDMMC_POWER.DIRPOL control bit.
## 60.8 SDMMC interrupts

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC</td>
<td>Command response CRC fail</td>
<td>CCRCFAIL</td>
<td>CCRCFAILIE</td>
<td>CCRCFAILC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data block CRC fail</td>
<td>DCRFAIL</td>
<td>DCRFAILIE</td>
<td>DCRFAILC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Command response timeout</td>
<td>CTIMEOUT</td>
<td>CTIMEOUTIE</td>
<td>CTIMEOUTC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data timeout</td>
<td>DTIMEOUT</td>
<td>DTIMEOUTIE</td>
<td>DTIMEOUTC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Transmit FIFO underrun</td>
<td>TXUNDER</td>
<td>TXUNDERIE</td>
<td>TXUNDERRC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Receive FIFO overrun</td>
<td>RXOVER</td>
<td>RXOVERIE</td>
<td>RXOVERRC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Command response received</td>
<td>CMDREND</td>
<td>CMDRENDIE</td>
<td>CMDRENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Command sent</td>
<td>CMDSENT</td>
<td>CMDSENTE</td>
<td>CMDSENTEC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data transfer ended</td>
<td>DATAEND</td>
<td>DATAENDIE</td>
<td>DATAENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data transfer hold</td>
<td>DHOLD</td>
<td>DHOLDIE</td>
<td>DHOLDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data block sent or received</td>
<td>DBCKEND</td>
<td>DBCKENDIE</td>
<td>DBCKENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data transfer aborted</td>
<td>DABORT</td>
<td>DABORTIE</td>
<td>DABORTC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Transmit FIFO half empty</td>
<td>TXFIHO</td>
<td>TXFIHOIE</td>
<td>n.a.</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Receive FIFO half full</td>
<td>RXFIHO</td>
<td>RXFIHOHIE</td>
<td>n.a.</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Transmit FIFO full</td>
<td>TXFIFOF</td>
<td>n.a.</td>
<td>n.a.</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Receive FIFO full</td>
<td>RXFIFOF</td>
<td>RXFIFOFIE</td>
<td>n.a.</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Transmit FIFO empty</td>
<td>TXFIFOE</td>
<td>TXFIFOEIE</td>
<td>n.a.</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Receive FIFO empty</td>
<td>RXFIFOE</td>
<td>n.a.</td>
<td>n.a.</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Table 456. SDMMC interrupts (continued)

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC</td>
<td>Command response end of busy</td>
<td>BUSYD0END</td>
<td>BUSYD0ENDIE</td>
<td>BUSYD0ENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>SDIO interrupt</td>
<td>SDIOIT</td>
<td>SDIOITIE</td>
<td>SDIOITC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Boot acknowledge ment fail</td>
<td>ACKFAIL</td>
<td>ACKFAILIE</td>
<td>ACKFAILC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Boot acknowledge ment timeout</td>
<td>ACKTIMEOUT</td>
<td>ACKTIMEOUTIE</td>
<td>ACKTIMEOUTC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Voltage switch timing</td>
<td>VSWEND</td>
<td>VSWENDIE</td>
<td>VSWENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>SDMM_CK stopped in voltage switch</td>
<td>CKSTOP</td>
<td>CKSTOPIE</td>
<td>CKSTOPC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>IDMA transfer error</td>
<td>IDMATE</td>
<td>IDMATEIE</td>
<td>IDMATEC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>IDMA buffer transfer complete</td>
<td>IDMABTC</td>
<td>IDMABTCIE</td>
<td>IDMABTCC</td>
<td>Yes</td>
</tr>
</tbody>
</table>
60.9 SDMMC registers

The device communicates to the system via 32-bit control registers accessible via AHB slave interface.

The peripheral registers have to be accessed by words (32-bit). Byte (8-bit) and halfword (16-bit) accesses trigger an AHB bus error.

60.9.1 SDMMC power control register (SDMMC_POWER)

Address offset: 0x000

Reset value: 0x0000 0000

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | rw | rw | rw |

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **DIRPOL**: Data and command direction signals polarity selection

This bit can only be written when the SDMMC is in the power-off state (PWRCTRL = 00).
0: Voltage transceiver IOs driven as output when direction signal is low.
1: Voltage transceiver IOs driven as output when direction signal is high.

Bit 3 **VSWITCHEN**: Voltage switch procedure enable

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
This bit is used to stop the SDMMC_CK after the voltage switch command response:
0: SDMMC_CK clock kept unchanged after successfully received command response.
1: SDMMC_CK clock stopped after successfully received command response.

Bit 2 **VSWITCH**: Voltage switch sequence start

This bit is used to start the timing critical section of the voltage switch sequence:
0: Voltage switch sequence not started and not active.
1: Voltage switch sequence started or active.

Bits 1:0 **PWRCTRL[1:0]**: SDMMC state control bits

These bits can only be written when the SDMMC is not in the power-on state (PWRCTRL ≠ 11).
These bits are used to define the functional state of the SDMMC signals:
00: After reset, Reset: the SDMMC is disabled and the clock to the Card is stopped, SDMMC_D[7:0], and SDMMC_CMD are HiZ and SDMMC_CK is driven low. When written 00, power-off: the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven high.
01: Reserved. (When written 01, PWRCTRL value will not change)
10: Power-cycle, the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low.
11: Power-on: the card is clocked. The first 74 SDMMC_CK cycles the SDMMC is still disabled. After the 74 cycles the SDMMC is enabled and the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are controlled according the SDMMC operation. Any further write will be ignored, PWRCTRL value will keep 11.
60.9.2 SDMMC clock control register (SDMMC_CLKCR)

Address offset: 0x004
Reset value: 0x0000 0000

The SDMMC_CLKCR register controls the SDMMC_CK output clock, the sdmmc_rx_ck receive clock, and the bus width.

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

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

Bits 21:20 **SELCLKRX[1:0]**: Receive clock selection
- These bits can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)
- 00: sdmmc_io_in_ck selected as receive clock
- 01: SDMMC_CKIN feedback clock selected as receive clock
- 10: sdmmc_fb_ck tuned feedback clock selected as receive clock.
- 11: Reserved (select sdmmc_io_in_ck)

Bit 19 **BUSSPEED**: Bus speed mode selection between DS, HS, SDR12, SDR25 and SDR50, DDR50, SDR104
- This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)
- 0: DS, HS, SDR12, SDR25 bus speed mode selected
- 1: SDR50, DDR50, SDR104 bus speed mode selected.

Bit 18 **DDR**: Data rate signaling selection
- This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)
- DDR rate shall only be selected with 4-bit or 8-bit wide bus mode. (WIDBUS > 00). DDR = 1 has no effect when WIDBUS = 00 (1-bit wide bus).
- DDR rate shall only be selected with clock division >1. (CLKDIV > 0)
- 0: SDR Single data rate signaling
- 1: DDR double data rate signaling

Bit 17 **HWFC_EN**: Hardware flow control enable
- This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)
- 0: Hardware flow control is disabled
- 1: Hardware flow control is enabled
- When Hardware flow control is enabled, the meaning of the TXFIFOE and RXFIFOF flags change, please see SDMMC status register definition in Section 60.9.11.
Bit 16 **NEGEDGE**: SDMMC\_CK dephasing selection bit for data and command

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

When clock division = 1 (CLKDIV = 0), this bit has no effect. Data and Command change on SDMMC\_CK falling edge.

0: When clock division > 1 (CLKDIV > 0) & DDR = 0:
- Command and data changed on the sdmcc\_ker\_ck falling edge succeeding the rising edge of SDMMC\_CK.
- SDMMC\_CK edge occurs on sdmcc\_ker\_ck rising edge.

When clock division > 1 (CLKDIV > 0) & DDR = 1:
- Command changed on the sdmcc\_ker\_ck falling edge succeeding the rising edge of SDMMC\_CK.
- Data changed on the sdmcc\_ker\_ck falling edge succeeding a SDMMC\_CK edge.
- SDMMC\_CK edge occurs on sdmcc\_ker\_ck rising edge.

1: When clock division > 1 (CLKDIV > 0) & DDR = 0:
- Command and data changed on the same sdmcc\_ker\_ck rising edge generating the SDMMC\_CK falling edge.

When clock division > 1 (CLKDIV > 0) & DDR = 1:
- Command changed on the same sdmcc\_ker\_ck rising edge generating the SDMMC\_CK falling edge.
- Data changed on the SDMMC\_CK falling edge succeeding a SDMMC\_CK edge.
- SDMMC\_CK edge occurs on sdmcc\_ker\_ck rising edge.

Bits 15:14 **WIDBUS[1:0]**: Wide bus mode enable bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: Default 1-bit wide bus mode: SDMMC\_D0 used (Does not support DDR)
01: 4-bit wide bus mode: SDMMC\_D[3:0] used
10: 8-bit wide bus mode: SDMMC\_D[7:0] used

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PWRSAV**: Power saving configuration bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

For power saving, the SDMMC\_CK clock output can be disabled when the bus is idle by setting PWRSAV:

0: SDMMC\_CK clock is always enabled
1: SDMMC\_CK is only enabled when the bus is active

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **CLKDIV[9:0]**: Clock divide factor

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

This field defines the divide factor between the input clock (sdmmc\_ker\_ck) and the output clock (SDMMC\_CK):

\[
SDMMC\_CK frequency = sdmmc\_ker\_ck / [2 * CLKDIV],
\]

0x000: SDMMC\_CK frequency = sdmmc\_ker\_ck / 1 (Does not support DDR)
0x001: SDMMC\_CK frequency = sdmmc\_ker\_ck / 2
0x002: SDMMC\_CK frequency = sdmmc\_ker\_ck / 4
0x00X: etc..
0x080: SDMMC\_CK frequency = sdmmc\_ker\_ck / 256
0x0XX: etc..
0x3FF: SDMMC\_CK frequency = sdmmc\_ker\_ck / 2046
Note:
1. While the SD/SDIO card or eMMC is in identification mode, the SDMMC_CK frequency must be less than 400 kHz.
2. The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.
3. At least seven sdmmc_hclk clock periods are needed between two write accesses to this register. SDMMC_CK can also be stopped during the ReadWait interval for SD I/O cards: in this case the SDMMC_CLKCR register does not control SDMMC_CK.

60.9.3 SDMMC argument register (SDMMC_ARGR)

Address offset: 0x008
Reset value: 0x0000 0000

The SDMMC_ARGR register contains a 32-bit command argument, which is sent to a card as part of a command message.

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

Bits 31:0 CMDARG[31:0]: Command argument
These bits can only be written by firmware when CPSM is disabled (CPSMEN = 0). Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

60.9.4 SDMMC command register (SDMMC_CMDR)

Address offset: 0x00C
Reset value: 0x0000 0000

The SDMMC_CMDR register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).

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

<table>
<thead>
<tr>
<th>BOOT EN</th>
<th>BOOT MODE</th>
<th>DT HOLD</th>
<th>CPSM EN</th>
<th>WAITP END</th>
<th>WAIT INT</th>
<th>WARTRESP[1:0]</th>
<th>CMD STOP</th>
<th>CMD TRANS</th>
<th>CMDINDEX[5:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bit 16  **CMD_SUSPEND**: The CPSM treats the command as a Suspend or Resume command and signals interrupt period start/end
- This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
- CMD_SUSPEND = 1 and CMD_TRANS = 0 Suspend command, start interrupt period when response bit BS=0.
- CMD_SUSPEND = 1 and CMD_TRANS = 1 Resume command with data, end interrupt period when response bit DF=1.

Bit 15  **BOOTEN**: Enable boot mode procedure
- 0: Boot mode procedure disabled
- 1: Boot mode procedure enabled

Bit 14  **BOOTMODE**: Select the boot mode procedure to be used
- This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0)
- 0: Normal boot mode procedure selected
- 1: Alternative boot mode procedure selected.

Bit 13  **DTHOLD**: Hold new data block transmission and reception in the DPSM
- If this bit is set, the DPSM will not move from the Wait_S state to the Send state or from the Wait_R state to the Receive state.

Bit 12  **CPSMEN**: Command path state machine (CPSM) enable bit
- This bit is written 1 by firmware, and cleared by hardware when the CPSM enters the Idle state.
- If this bit is set, the CPSM is enabled.
- When DTEN = 1, no command will be transferred nor boot procedure will be started.
- CPSMEN is cleared to 0.
- During ReadWait with SDMMC_CK stopped no command will be sent and CPSMEN is kept 0.

Bit 11  **WAITPEND**: CPSM waits for end of data transfer (CmdPend internal signal) from DPSM
- This bit when set, the CPSM waits for the end of data transfer trigger before it starts sending a command.
- WAITPEND is only taken into account when DT_MODE = eMMC stream data transfer,
  WIDBUS = 1-bit wide bus mode, DPSMACT = 1 and DT_DIR = from host to card.

Bit 10  **WAITINT**: CPSM waits for interrupt request
- If this bit is set, the CPSM disables command timeout and waits for an card interrupt request (Response).
- If this bit is cleared in the CPSM Wait state, will cause the abort of the interrupt mode.

Bits 9:8  **WAITRESP[1:0]**: Wait for response bits
- This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
- They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.
- 00: No response, expect CMDSENT flag
- 01: Short response, expect CMDREND or CCRCFAIL flag
- 10: Short response, expect CMDREND flag (No CRC)
- 11: Long response, expect CMDREND or CCRCFAIL flag
Bit 7 **CMDSTOP**: The CPSM treats the command as a Stop Transmission command and signals Abort to the DPSM  
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).  
If this bit is set, the CPSM issues the Abort signal to the DPSM when the command is sent.

Bit 6 **CMDTRANS**: The CPSM treats the command as a data transfer command, stops the interrupt period, and signals DataEnable to the DPSM  
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).  
If this bit is set, the CPSM issues an end of interrupt period and issues DataEnable signal to the DPSM when the command is sent.

Bits 5:0 **CMDINDEX[5:0]**: Command index  
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).  
The command index is sent to the card as part of a command message.

**Note:**  
1. At least seven sdmmc_hclk clock periods are needed between two write accesses to this register.  
2. MultiMediaCard can send two kinds of response: short responses, 48 bits, or long responses, 136 bits. SD card and SD I/O card can send only short responses, the argument can vary according to the type of response: the software will distinguish the type of response according to the send command.

**60.9.5 SDMMC command response register (SDMMC_RESPCMDR)**

Address offset: 0x010  
Reset value: 0x0000 0000

The SDMMC_RESPCMDR register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

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

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RESPCMD[5:0]**: Response command index  
Read-only bit field. Contains the command index of the last command response received.
60.9.6 SDMMC response x register (SDMMC_RESPxR)

Address offset: \(0x010 + 0x004 \times x\), (\(x = 1\) to 4)
Reset value: 0x0000 0000

The SDMMC_RESP1/2/3/4R registers contain the status of a card, which is part of the received response.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>CARDSTATUSx[31:0]: see Table 457</th>
</tr>
</thead>
</table>

The card status size is 32 or 128 bits, depending on the response type.

Table 457. Response type and SDMMC_RESPxR registers

<table>
<thead>
<tr>
<th>Register(1)</th>
<th>Short response</th>
<th>Long response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC_RESP1R</td>
<td>Card status[31:0]</td>
<td>Card status [127:96]</td>
</tr>
<tr>
<td>SDMMC_RESP2R</td>
<td>all 0</td>
<td>Card status [95:64]</td>
</tr>
<tr>
<td>SDMMC_RESP3R</td>
<td>all 0</td>
<td>Card status [63:32]</td>
</tr>
<tr>
<td>SDMMC_RESP4R</td>
<td>all 0</td>
<td>Card status <a href="2">31:0</a></td>
</tr>
</tbody>
</table>

1. The most significant bit of the card status is received first.
2. The SDMMC_RESP4R register LSB is always 0.

60.9.7 SDMMC data timer register (SDMMC_DTIMER)

Address offset: 0x024
Reset value: 0x0000 0000

The SDMMC_DTIMER register contains the data timeout period, in card bus clock periods.

A counter loads the value from the SDMMC_DTIMER register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>DATATIME[31:16]:</th>
<th>DATATIME[15:0]:</th>
</tr>
</thead>
</table>

---

2378/2974 RM0455 Rev 3
Note: A data transfer must be written to the data timer register and the data length register before being written to the data control register.

60.9.8 SDMMC data length register (SDMMC_DLENR)

Address offset: 0x028

Reset value: 0x0000 0000

The SDMMC_DLENR register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

<table>
<thead>
<tr>
<th>Bits 31:25</th>
<th>Bits 24:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>DATALENGTH[24:0]: Data length value</td>
</tr>
<tr>
<td>31:25</td>
<td>24:0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 DATALENGTH[24:0]: Data length value

This register can only be written by firmware when DPSM is inactive (DPSMCTL = 0).

Number of data bytes to be transferred.

When DDR = 1 DATALENGTH is truncated to a multiple of 2. (The last odd byte is not transferred)

When DATALENGTH = 0 no data will be transferred, when requested by a CPSMEN and CMDTRANS = 1 also no command will be transferred. DTEN and CPSMEN are cleared to 0.

Note: For a block data transfer, the value in the data length register must be a multiple of the block size (see SDMMC_DCTRL). A data transfer must be written to the data timer register and the data length register before being written to the data control register.

For an SDMMC multibyte transfer the value in the data length register must be between 1 and 512.
### 60.9.9 SDMMC data control register (SDMMC_DCTRL)

Address offset: 0x02C  
Reset value: 0x0000 0000  

The SDMMC_DCTRL register control the data path state machine (DPSM).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>FIFO RST</th>
<th>BOOT ACK</th>
<th>SDIO EN</th>
<th>RW MOD</th>
<th>RW STOP</th>
<th>RW START</th>
<th>DBLOCKSIZE[3:0]</th>
<th>DMODE[1:0]</th>
<th>DTDIR</th>
<th>DTEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

- Bits 31:14 Reserved, must be kept at reset value.
- **Bit 13 FIFO_RST**: FIFO reset, will flush any remaining data  
  This bit can only be written by firmware when IDMAEN = 0 and DPSM is active (DPSMACT = 1). This bit will only take effect when a transfer error or transfer hold occurs.  
  - 0: FIFO not affected.  
  - 1: Flush any remaining data and reset the FIFO pointers. This bit automatically will be cleared to 0 by hardware when DPSM gets inactive (DPSMACT = 0).
- **Bit 12 BOOTACKEN**: Enable the reception of the boot acknowledgment  
  This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).  
  - 0: Boot acknowledgment disabled, not expected to be received  
  - 1: Boot acknowledgment enabled, expected to be received
- **Bit 11 SDIOEN**: SD I/O interrupt enable functions  
  This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). If this bit is set, the DPSM enables the SD I/O card specific interrupt operation.
- **Bit 10 RWMOD**: Read wait mode  
  This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).  
  - 0: Read Wait control using SDMMC_D2  
  - 1: Read Wait control stopping SDMMC_CK
- **Bit 9 RWSTOP**: Read wait stop  
  This bit is written by firmware and auto cleared by hardware when the DPSM moves from the READ_WAIT state to the WAIT_R or IDLE state.  
  - 0: No read wait stop  
  - 1: Enable for read wait stop when DPSM is in the READ_WAIT state
- **Bit 8 RWSTART**: Read wait start  
  If this bit is set, read wait operation starts.
Bits 7:4 **DBLOCKSIZE[3:0]**: Data block size

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Define the data block length when the block data transfer mode is selected:

- 0000: (0 decimal) lock length = $2^0 = 1$ byte
- 0001: (1 decimal) lock length = $2^1 = 2$ bytes
- 0010: (2 decimal) lock length = $2^2 = 4$ bytes
- 0011: (3 decimal) lock length = $2^3 = 8$ bytes
- 0100: (4 decimal) lock length = $2^4 = 16$ bytes
- 0101: (5 decimal) lock length = $2^5 = 32$ bytes
- 0110: (6 decimal) lock length = $2^6 = 64$ bytes
- 0111: (7 decimal) lock length = $2^7 = 128$ bytes
- 1000: (8 decimal) lock length = $2^8 = 256$ bytes
- 1001: (9 decimal) lock length = $2^9 = 512$ bytes
- 1010: (10 decimal) lock length = $2^{10} = 1024$ bytes
- 1011: (11 decimal) lock length = $2^{11} = 2048$ bytes
- 1100: (12 decimal) lock length = $2^{12} = 4096$ bytes
- 1101: (13 decimal) lock length = $2^{13} = 8192$ bytes
- 1110: (14 decimal) lock length = $2^{14} = 16384$ bytes
- 1111: (15 decimal) reserved

When DATALENGTH is not a multiple of DBLOCKSIZE, the transferred data is truncated at a multiple of DBLOCKSIZE. (Any remain data will not be transferred.)

When DDR = 1, DBLOCKSIZE = 0000 shall not be used. (No data will be transferred)

Bits 3:2 **DTMODE[1:0]**: Data transfer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

- 00: Block data transfer ending on block count.
- 01: SDIO multibyte data transfer.
- 10: e-MMC Stream data transfer. (WIDBUS shall select 1-bit wide bus mode)
- 11: Block data transfer ending with STOP_TRANSMISSION command (not to be used with DTEN initiated data transfers).

Bit 1 **DTDIR**: Data transfer direction selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

- 0: From host to card.
- 1: From card to host.

Bit 0 **DTEN**: Data transfer enable bit

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). This bit is cleared by Hardware when data transfer completes. This bit shall only be used to transfer data when no associated data transfer command is used, i.e. shall not be used with SD or e-MMC cards.

- 0: Do not start data transfer without CPSM data transfer command.
- 1: Start data transfer without CPSM data transfer command.

### 60.9.10 SDMMC data counter register (SDMMC_DCNTR)

Address offset: 0x030

Reset value: 0x0000 0000

The SDMMC_DCNTR register loads the value from the data length register (see SDMMC_DLENR) when the DPSM moves from the Idle state to the Wait_R or Wait_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then
moves to the Idle state and when there has been no error, and no transmit data transfer hold, the data status end flag (DATAEND) is set.

Note: This register should be read only after the data transfer is complete, or hold. When reading after an error event the read data count value may be different from the real number of data bytes transferred.

### 60.9.11 SDMMC status register (SDMMC_STAR)

Address offset: 0x034

Reset value: 0x0000 0000

The SDMMC_STAR register is a read-only register. It contains two types of flag:
- Static flags (bits [28, 21, 11:0]): these bits remain asserted until they are cleared by writing to the SDMMC interrupt Clear register (see SDMMC_ICR)
- Dynamic flags (bits [20:12]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO)

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATACOUNT[24:0]:** Data count value

When read, the number of remaining data bytes to be transferred is returned. Write has no effect.

**Note:** This register should be read only after the data transfer is complete, or hold. When reading after an error event the read data count value may be different from the real number of data bytes transferred.

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

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

Bits 31:29 Reserved, must be kept at reset value.

- **Bit 28** IDMABTC: IDMA buffer transfer complete
  The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

- **Bit 27** IDMATE: IDMA transfer error
  The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

- **Bit 26** CKSTOP: SDMMC_CK stopped in Voltage switch procedure
  The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
Bit 25 **VSWEND**: Voltage switch critical timing section completion
   The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 24 **ACKTIMEOUT**: Boot acknowledgment timeout
   The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 23 **ACKFAIL**: Boot acknowledgment received (boot acknowledgment check fail)
   The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 22 **SDIOIT**: SDIO interrupt received
   The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 21 **BUSYD0END**: end of SDMMC_D0 Busy following a CMD response detected
   This indicates only end of busy following a CMD response. This bit does not signal busy due to data transfer. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
   - 0: card SDMMC_D0 signal does NOT signal change from busy to not busy.
   - 1: card SDMMC_D0 signal changed from busy to NOT busy.

Bit 20 **BUSYD0**: Inverted value of SDMMC_D0 line (Busy), sampled at the end of a CMD response and a second time 2 SDMMC_CK cycles after the CMD response
   This bit is reset to not busy when the SDMMCD0 line changes from busy to not busy. This bit does not signal busy due to data transfer. This is a hardware status flag only, it does not generate an interrupt.
   - 0: card signals not busy on SDMMC_D0.
   - 1: card signals busy on SDMMC_D0.

Bit 19 **RXFIFOE**: Receive FIFO empty
   This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes full.

Bit 18 **TXFIFOE**: Transmit FIFO empty
   This bit is cleared when one FIFO location becomes full.

Bit 17 **RXFIFOF**: Receive FIFO full
   This bit is cleared when one FIFO location becomes empty.

Bit 16 **TXFIFOF**: Transmit FIFO full
   This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes empty.

Bit 15 **RXFIFOHF**: Receive FIFO half full
   There are at least half the number of words in the FIFO. This bit is cleared when the FIFO becomes half+1 empty.

Bit 14 **TXFIFOHE**: Transmit FIFO half empty
   At least half the number of words can be written into the FIFO. This bit is cleared when the FIFO becomes half+1 full.

Bit 13 **CPSMACT**: Command path state machine active, i.e. not in Idle state
   This is a hardware status flag only, does not generate an interrupt.

Bit 12 **DPSMACT**: Data path state machine active, i.e. not in Idle state
   This is a hardware status flag only, does not generate an interrupt.

Bit 11 **DABORT**: Data transfer aborted by CMD12
   Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td><strong>DBCKEND</strong>: Data block sent/received</td>
<td>DBCKEND is set when:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- CRC check passed and DPSM moves to the READWAIT state</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IDMAEN = 0 and transmit data transfer hold and DATACOUNT &gt;0 and DPSM moves to WAIT_S.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.</td>
</tr>
<tr>
<td>9</td>
<td><strong>D HOLD</strong>: Data transfer Hold</td>
<td>Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.</td>
</tr>
<tr>
<td>8</td>
<td><strong>DATAEND</strong>: Data transfer ended correctly</td>
<td>DATAEND is set if data counter DATACOUNT is zero and no errors occur, and no transmit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>data transfer hold.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.</td>
</tr>
<tr>
<td>7</td>
<td><strong>CMDSENT</strong>: Command sent (no response required)</td>
<td>Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.</td>
</tr>
<tr>
<td>6</td>
<td><strong>CMDREND</strong>: Command response received (CRC check passed, or no CRC)</td>
<td>Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.</td>
</tr>
<tr>
<td>5</td>
<td><strong>RXOVERR</strong>: Received FIFO overrun error</td>
<td>Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.</td>
</tr>
<tr>
<td>4</td>
<td><strong>TXUNDERR</strong>: Transmit FIFO underrun error</td>
<td>Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.</td>
</tr>
<tr>
<td>3</td>
<td><strong>DTIMEOUT</strong>: Data timeout</td>
<td>Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.</td>
</tr>
<tr>
<td>2</td>
<td><strong>CTIMEOUT</strong>: Command response timeout</td>
<td>Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Command Timeout period has a fixed value of 64 SDMMC_CK clock periods.</td>
</tr>
<tr>
<td>1</td>
<td><strong>DCRCFAIL</strong>: Data block sent/received (CRC check failed)</td>
<td>Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.</td>
</tr>
<tr>
<td>0</td>
<td><strong>CCRCFAIL</strong>: Command response received (CRC check failed)</td>
<td>Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.</td>
</tr>
</tbody>
</table>

*Note:* FIFO interrupt flags shall be masked in SDMMC_MASKR when using IDMA mode.
60.9.12 SDMMC interrupt clear register (SDMMC_ICR)

Address offset: 0x038

Reset value: 0x0000 0000

The SDMMC_ICR register is a write-only register. Writing a bit with 1 clears the corresponding bit in the SDMMC_STAR status register.

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:29 Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28 IDMABTCC</td>
<td>IDMA buffer transfer complete clear bit</td>
</tr>
<tr>
<td>27 IDMATEC</td>
<td>IDMA transfer error clear bit</td>
</tr>
<tr>
<td>26 CKSTOPC</td>
<td>CKSTOP flag clear bit</td>
</tr>
<tr>
<td>25 VSWENDC</td>
<td>VSWEND flag clear bit</td>
</tr>
<tr>
<td>24 ACKTIMEOUTC</td>
<td>ACKTIMEOUT flag clear bit</td>
</tr>
<tr>
<td>23 ACKFAILC</td>
<td>ACKFAIL flag clear bit</td>
</tr>
<tr>
<td>22 SDIOITC</td>
<td>SDIOIT flag clear bit</td>
</tr>
</tbody>
</table>

Bit 28 IDMABTCC: IDMA buffer transfer complete clear bit
Set by software to clear the IDMABTC flag.
0: IDMABTC not cleared
1: IDMABTC cleared

Bit 27 IDMATEC: IDMA transfer error clear bit
Set by software to clear the IDMATE flag.
0: IDMATE not cleared
1: IDMATE cleared

Bit 26 CKSTOPC: CKSTOP flag clear bit
Set by software to clear the CKSTOP flag.
0: CKSTOP not cleared
1: CKSTOP cleared

Bit 25 VSWENDC: VSWEND flag clear bit
Set by software to clear the VSWEND flag.
0: VSWEND not cleared
1: VSWEND cleared

Bit 24 ACKTIMEOUTC: ACKTIMEOUT flag clear bit
Set by software to clear the ACKTIMEOUT flag.
0: ACKTIMEOUT not cleared
1: ACKTIMEOUT cleared

Bit 23 ACKFAILC: ACKFAIL flag clear bit
Set by software to clear the ACKFAIL flag.
0: ACKFAIL not cleared
1: ACKFAIL cleared

Bit 22 SDIOITC: SDIOIT flag clear bit
Set by software to clear the SDIOIT flag.
0: SDIOIT not cleared
1: SDIOIT cleared
Bit 21 **BUSYD0ENDC**: BUSYD0END flag clear bit
   Set by software to clear the BUSYD0END flag.
   0: BUSYD0END not cleared
   1: BUSYD0END cleared

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **DABORTC**: DABORT flag clear bit
   Set by software to clear the DABORT flag.
   0: DABORT not cleared
   1: DABORT cleared

Bit 10 **DBCKENDC**: DBCKEND flag clear bit
   Set by software to clear the DBCKEND flag.
   0: DBCKEND not cleared
   1: DBCKEND cleared

Bit 9 **DHOLODC**: DHOLOD flag clear bit
   Set by software to clear the DHOLOD flag.
   0: DHOLOD not cleared
   1: DHOLOD cleared

Bit 8 **DATAENDC**: DATAEND flag clear bit
   Set by software to clear the DATAEND flag.
   0: DATAEND not cleared
   1: DATAEND cleared

Bit 7 **CMDSENTECT**: CMDSENT flag clear bit
   Set by software to clear the CMDSENT flag.
   0: CMDSENT not cleared
   1: CMDSENT cleared

Bit 6 **CMDRENDENC**: CMDRENDE flag clear bit
   Set by software to clear the CMDRENDE flag.
   0: CMDRENDE not cleared
   1: CMDRENDE cleared

Bit 5 **RXOVERRC**: RXOVERR flag clear bit
   Set by software to clear the RXOVERR flag.
   0: RXOVERR not cleared
   1: RXOVERR cleared

Bit 4 **TXUNDERRC**: TXUNDERR flag clear bit
   Set by software to clear the TXUNDERR flag.
   0: TXUNDERR not cleared
   1: TXUNDERR cleared

Bit 3 **DTIMEOUTC**: DTIMEOUT flag clear bit
   Set by software to clear the DTIMEOUT flag.
   0: DTIMEOUT not cleared
   1: DTIMEOUT cleared
Bit 2 **CTIMEOUT**: CTIMEOUT flag clear bit
   Set by software to clear the CTIMEOUT flag.
   0: CTIMEOUT not cleared
   1: CTIMEOUT cleared

Bit 1 **DCRCFAIL**: DCRCFAIL flag clear bit
   Set by software to clear the DCRCFAIL flag.
   0: DCRCFAIL not cleared
   1: DCRCFAIL cleared

Bit 0 **CCRCFAIL**: CCRCFAIL flag clear bit
   Set by software to clear the CCRCFAIL flag.
   0: CCRCFAIL not cleared
   1: CCRCFAIL cleared

### 60.9.13 SDMMC mask register (SDMMC_MASKR)

Address offset: 0x03C

Reset value: 0x0000 0000

The interrupt mask register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RXFIE</td>
<td>TXFIE</td>
<td>Res.</td>
<td>DA</td>
<td>BORTIE</td>
<td>DBCKIE</td>
<td>ENDIE</td>
<td>DHOLDIE</td>
<td>DATAIE</td>
<td>CMDSENIE</td>
<td>CMDRIE</td>
<td>RXOVERRIE</td>
<td>TXUNDERIE</td>
<td>DTIMEOUT</td>
<td>DCRCFAIL</td>
<td>CCRCFAIL</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTIE**: IDMA buffer transfer complete interrupt enable
   Set and cleared by software to enable/disable the interrupt generated when the IDMA has transferred all data belonging to a memory buffer.
   0: IDMA buffer transfer complete interrupt disabled
   1: IDMA buffer transfer complete interrupt enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CKSTOPIE**: Voltage Switch clock stopped interrupt enable
   Set and cleared by software to enable/disable interrupt caused by Voltage Switch clock stopped.
   0: Voltage Switch clock stopped interrupt disabled
   1: Voltage Switch clock stopped interrupt enabled

Bit 25 **VSWENDIE**: Voltage switch critical timing section completion interrupt enable
   Set and cleared by software to enable/disable the interrupt generated when voltage switch critical timing section completion.
   0: Voltage switch critical timing section completion interrupt disabled
   1: Voltage switch critical timing section completion interrupt enabled
Secure digital input/output MultiMediaCard interface (SDMMC)  

Bit 24 **ACKTIMEOUTIE**: Acknowledgment timeout interrupt enable  
Set and cleared by software to enable/disable interrupt caused by acknowledgment timeout.  
0: Acknowledgment timeout interrupt disabled  
1: Acknowledgment timeout interrupt enabled

Bit 23 **ACKFAILIE**: Acknowledgment Fail interrupt enable  
Set and cleared by software to enable/disable interrupt caused by acknowledgment Fail.  
0: Acknowledgment Fail interrupt disabled  
1: Acknowledgment Fail interrupt enabled

Bit 22 **SDIOITIE**: SDIO mode interrupt received interrupt enable  
Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.  
0: SDIO Mode interrupt received interrupt disabled  
1: SDIO Mode interrupt received interrupt enabled

Bit 21 **BUSYD0ENDIE**: BUSYD0END interrupt enable  
Set and cleared by software to enable/disable the interrupt generated when SDMMC_D0 signal changes from busy to NOT busy following a CMD response.  
0: BUSYD0END interrupt disabled  
1: BUSYD0END interrupt enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TXFIFOEIE**: Tx FIFO empty interrupt enable  
Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.  
0: Tx FIFO empty interrupt disabled  
1: Tx FIFO empty interrupt enabled

Bit 17 **RXFIFOFIE**: Rx FIFO full interrupt enable  
Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.  
0: Rx FIFO full interrupt disabled  
1: Rx FIFO full interrupt enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **RXFIFOHFIE**: Rx FIFO half full interrupt enable  
Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.  
0: Rx FIFO half full interrupt disabled  
1: Rx FIFO half full interrupt enabled

Bit 14 **TXFIFOHEIE**: Tx FIFO half empty interrupt enable  
Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.  
0: Tx FIFO half empty interrupt disabled  
1: Tx FIFO half empty interrupt enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **DABORTIE**: Data transfer aborted interrupt enable  
Set and cleared by software to enable/disable interrupt caused by a data transfer being aborted.  
0: Data transfer abort interrupt disabled  
1: Data transfer abort interrupt enabled

Bit 10 **DBCKENDIE**: Data block end interrupt enable  
Set and cleared by software to enable/disable interrupt caused by data block end.  
0: Data block end interrupt disabled  
1: Data block end interrupt enabled
Bit 9  **DHOLDIE**: Data hold interrupt enable  
Set and cleared by software to enable/disable the interrupt generated when sending new data is held in the DPSM Wait_S state.  
0: Data hold interrupt disabled  
1: Data hold interrupt enabled  

Bit 8  **DATAENDIE**: Data end interrupt enable  
Set and cleared by software to enable/disable interrupt caused by data end.  
0: Data end interrupt disabled  
1: Data end interrupt enabled  

Bit 7  **CMDSENTIE**: Command sent interrupt enable  
Set and cleared by software to enable/disable interrupt caused by sending command.  
0: Command sent interrupt disabled  
1: Command sent interrupt enabled  

Bit 6  **CMDRENDIE**: Command response received interrupt enable  
Set and cleared by software to enable/disable interrupt caused by receiving command response.  
0: Command response received interrupt disabled  
1: Command response received interrupt enabled  

Bit 5  **RXOVERRIE**: Rx FIFO overrun error interrupt enable  
Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.  
0: Rx FIFO overrun error interrupt disabled  
1: Rx FIFO overrun error interrupt enabled  

Bit 4  **TXUNDERIE**: Tx FIFO underrun error interrupt enable  
Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.  
0: Tx FIFO underrun error interrupt disabled  
1: Tx FIFO underrun error interrupt enabled  

Bit 3  **DTIMEOUTIE**: Data timeout interrupt enable  
Set and cleared by software to enable/disable interrupt caused by data timeout.  
0: Data timeout interrupt disabled  
1: Data timeout interrupt enabled  

Bit 2  **CTIMEOUTIE**: Command timeout interrupt enable  
Set and cleared by software to enable/disable interrupt caused by command timeout.  
0: Command timeout interrupt disabled  
1: Command timeout interrupt enabled  

Bit 1  **DCRCFAILIE**: Data CRC fail interrupt enable  
Set and cleared by software to enable/disable interrupt caused by data CRC failure.  
0: Data CRC fail interrupt disabled  
1: Data CRC fail interrupt enabled  

Bit 0  **CCRCFAILIE**: Command CRC fail interrupt enable  
Set and cleared by software to enable/disable interrupt caused by command CRC failure.  
0: Command CRC fail interrupt disabled  
1: Command CRC fail interrupt enabled
60.9.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER)

Address offset: 0x040  
Reset value: 0x0000 0000

The SDMMC_ACKTIMER register contains the acknowledgment timeout period, in SDMMC_CK bus clock periods.

A counter loads the value from the SDMMC_ACKTIMER register, and starts decrementing when the data path state machine (DPSM) enters the Wait_Ack state. If the timer reaches 0 while the DPSM is in this states, the acknowledgment timeout status flag is set.

Note: The data transfer must be written to the acknowledgment timer register before being written to the data control register.

60.9.15 SDMMC data FIFO registers x (SDMMC_FIFORx)

Address offset: 0x080 + 0x004 * x, (x = 0 to 15)  
Reset value: 0x0000 0000  

The receive and transmit FIFOs can be only read or written as word (32-bit) wide registers. The FIFOs contain 16 entries on sequential addresses. This allows the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.
60.9.16 SDMMC DMA control register (SDMMC_IDMACTRLR)

Address offset: 0x050

Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This allows the CPU to use its load and store multiple operands to read from/write to the FIFO.

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

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

Bit 2 **IDMABACT**: Double buffer mode active buffer indication

- This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). When IDMA is enabled this bit is toggled by hardware.
- 0: When IDMA is enabled, uses buffer0 and firmware write access to IDMABASE0 is prohibited.
- 1: When IDMA is enabled, uses buffer1 and firmware write access to IDMABASE1 is prohibited.

Bit 1 **IDMABMODE**: Buffer mode selection

- This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
- 0: Single buffer mode.
- 1: Double buffer mode.

Bit 0 **IDMAEN**: IDMA enable

- This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
- 0: IDMA disabled
- 1: IDMA enabled
60.9.17  **SDMMC IDMA buffer size register (SDMMC_IDMABSIZER)**

Address offset: 0x054  
Reset value: 0x0000 0000  
The SDMMC_IDMABSIZER register contains the buffers size when in double buffer configuration.

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

![Register Diagram]

Bits 31:13  Reserved, must be kept at reset value.

Bits 12:5  **IDMABNDT[7:0]: Number of bytes per buffer**  
This 8-bit value shall be multiplied by 8 to get the size of the buffer in 32-bit words and by 32 to get the size of the buffer in bytes.
Example: IDMABNDT = 0x01: buffer size = 8 words = 32 bytes.
Example: IDMABNDT = 0x80: buffer size = 1024 words = 4 Kbyte
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 4:0  Reserved, must be kept at reset value.

60.9.18  **SDMMC IDMA buffer 0 base address register (SDMMC_IDMABASE0R)**

Address offset: 0x058  
Reset value: 0x0000 0000  
The SDMMC_IDMABASE0R register contains the memory buffer base address in single buffer configuration and the buffer 0 base address in double buffer configuration.

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

![Register Diagram]

Bits 31:0  **IDMABASE0[31:0]: Buffer 0 memory base address bits [31:2], shall be word aligned (bit [1:0] are always 0 and read only)**
This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1) and memory buffer 0 is inactive (IDMABACT = ‘1’).
60.9.19  SDMMC IDMA buffer 1 base address register (SDMMC_IDMABASE1R)

Address offset: 0x05C
Reset value: 0x0000 0000

The SDMMC_IDMABASE1R register contains the double buffer configuration second buffer memory base address.

Ranges:
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16          
IDMABASE[31:16]
            rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw           

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

Bits 31:0  IDMABASE1[31:0]: Buffer 1 memory base address, shall be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1) and memory buffer 1 is inactive (IDMABACT = '0').
### 60.9.20 SDMMC register map

The following table summarizes the SDMMC registers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset name</th>
<th>Name</th>
<th>Offset name</th>
<th>Name</th>
<th>Offset name</th>
<th>Name</th>
<th>Offset name</th>
<th>Name</th>
<th>Offset name</th>
<th>Name</th>
<th>Offset name</th>
<th>Name</th>
<th>Offset name</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>SDMMC_POWER</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>SDMMC_CLKCR</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>SDMMC_ARGR</td>
<td>CMDARG[31:0]</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C</td>
<td>SDMMC_CMDR</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>SDMMC_RESPCMDR</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>SDMMC_RESP1R</td>
<td>CARDSTATUS1[31:0]</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x018</td>
<td>SDMMC_RESP2R</td>
<td>CARDSTATUS2[31:0]</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01C</td>
<td>SDMMC_RESP3R</td>
<td>CARDSTATUS3[31:0]</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x020</td>
<td>SDMMC_RESP4R</td>
<td>CARDSTATUS4[31:0]</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x024</td>
<td>SDMMC_DTMR</td>
<td>DATETIME[31:0]</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x028</td>
<td>SDMMC_DLENR</td>
<td>DATALENGTH[24:0]</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 458. SDMMC register map (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02C</td>
<td>SDMMC_DCTRLR</td>
<td>0x02C</td>
<td>SDMMC_DCTRLR</td>
<td>0x02C</td>
<td>SDMMC_DCTRLR</td>
<td>0x02C</td>
<td>SDMMC_DCTRLR</td>
<td>0x02C</td>
<td>SDMMC_DCTRLR</td>
<td>0x02C</td>
<td>SDMMC_DCTRLR</td>
<td>0x02C</td>
<td>SDMMC_DCTRLR</td>
<td>0x02C</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x030</td>
<td>SDMMC_DCNTR</td>
<td>0x030</td>
<td>SDMMC_DCNTR</td>
<td>0x030</td>
<td>SDMMC_DCNTR</td>
<td>0x030</td>
<td>SDMMC_DCNTR</td>
<td>0x030</td>
<td>SDMMC_DCNTR</td>
<td>0x030</td>
<td>SDMMC_DCNTR</td>
<td>0x030</td>
<td>SDMMC_DCNTR</td>
<td>0x030</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x034</td>
<td>SDMMC_STAR</td>
<td>0x034</td>
<td>SDMMC_STAR</td>
<td>0x034</td>
<td>SDMMC_STAR</td>
<td>0x034</td>
<td>SDMMC_STAR</td>
<td>0x034</td>
<td>SDMMC_STAR</td>
<td>0x034</td>
<td>SDMMC_STAR</td>
<td>0x034</td>
<td>SDMMC_STAR</td>
<td>0x034</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x038</td>
<td>SDMMC_ICR</td>
<td>0x038</td>
<td>SDMMC_ICR</td>
<td>0x038</td>
<td>SDMMC_ICR</td>
<td>0x038</td>
<td>SDMMC_ICR</td>
<td>0x038</td>
<td>SDMMC_ICR</td>
<td>0x038</td>
<td>SDMMC_ICR</td>
<td>0x038</td>
<td>SDMMC_ICR</td>
<td>0x038</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x03C</td>
<td>SDMMC_MASKR</td>
<td>0x03C</td>
<td>SDMMC_MASKR</td>
<td>0x03C</td>
<td>SDMMC_MASKR</td>
<td>0x03C</td>
<td>SDMMC_MASKR</td>
<td>0x03C</td>
<td>SDMMC_MASKR</td>
<td>0x03C</td>
<td>SDMMC_MASKR</td>
<td>0x03C</td>
<td>SDMMC_MASKR</td>
<td>0x03C</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x040</td>
<td>SDMMC_ACKTIMER</td>
<td>0x040</td>
<td>SDMMC_ACKTIMER</td>
<td>0x040</td>
<td>SDMMC_ACKTIMER</td>
<td>0x040</td>
<td>SDMMC_ACKTIMER</td>
<td>0x040</td>
<td>SDMMC_ACKTIMER</td>
<td>0x040</td>
<td>SDMMC_ACKTIMER</td>
<td>0x040</td>
<td>SDMMC_ACKTIMER</td>
<td>0x040</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x044</td>
<td>Reserved</td>
<td>0x044</td>
<td>Reserved</td>
<td>0x044</td>
<td>Reserved</td>
<td>0x044</td>
<td>Reserved</td>
<td>0x044</td>
<td>Reserved</td>
<td>0x044</td>
<td>Reserved</td>
<td>0x044</td>
<td>Reserved</td>
<td>0x044</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x050</td>
<td>SDMMC_IDMACTRLR</td>
<td>0x050</td>
<td>SDMMC_IDMACTRLR</td>
<td>0x050</td>
<td>SDMMC_IDMACTRLR</td>
<td>0x050</td>
<td>SDMMC_IDMACTRLR</td>
<td>0x050</td>
<td>SDMMC_IDMACTRLR</td>
<td>0x050</td>
<td>SDMMC_IDMACTRLR</td>
<td>0x050</td>
<td>SDMMC_IDMACTRLR</td>
<td>0x050</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x054</td>
<td>SDMMC_IDMABSZ</td>
<td>0x054</td>
<td>SDMMC_IDMABSZ</td>
<td>0x054</td>
<td>SDMMC_IDMABSZ</td>
<td>0x054</td>
<td>SDMMC_IDMABSZ</td>
<td>0x054</td>
<td>SDMMC_IDMABSZ</td>
<td>0x054</td>
<td>SDMMC_IDMABSZ</td>
<td>0x054</td>
<td>SDMMC_IDMABSZ</td>
<td>0x054</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x058</td>
<td>SDMMC_IDMABASE0R</td>
<td>0x058</td>
<td>SDMMC_IDMABASE0R</td>
<td>0x058</td>
<td>SDMMC_IDMABASE0R</td>
<td>0x058</td>
<td>SDMMC_IDMABASE0R</td>
<td>0x058</td>
<td>SDMMC_IDMABASE0R</td>
<td>0x058</td>
<td>SDMMC_IDMABASE0R</td>
<td>0x058</td>
<td>SDMMC_IDMABASE0R</td>
<td>0x058</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x05C</td>
<td>SDMMC_IDMABASE1R</td>
<td>0x05C</td>
<td>SDMMC_IDMABASE1R</td>
<td>0x05C</td>
<td>SDMMC_IDMABASE1R</td>
<td>0x05C</td>
<td>SDMMC_IDMABASE1R</td>
<td>0x05C</td>
<td>SDMMC_IDMABASE1R</td>
<td>0x05C</td>
<td>SDMMC_IDMABASE1R</td>
<td>0x05C</td>
<td>SDMMC_IDMABASE1R</td>
<td>0x05C</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x060</td>
<td>Reserved</td>
<td>0x060</td>
<td>Reserved</td>
<td>0x060</td>
<td>Reserved</td>
<td>0x060</td>
<td>Reserved</td>
<td>0x060</td>
<td>Reserved</td>
<td>0x060</td>
<td>Reserved</td>
<td>0x060</td>
<td>Reserved</td>
<td>0x060</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>
Refer to Section 2.3 on page 128 for the register boundary addresses.
Controller area network with flexible data rate (FDCAN)

61.1 Introduction

The controller area network (CAN) subsystem (see Figure 698) consists of two CAN modules, a shared message RAM and a clock calibration unit. Refer to the product memory organization for the base address of each of them.

FDCAN modules are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

In addition, the first CAN module FDCAN1 supports time triggered CAN (TTCAN), specified in ISO 11898-4, including event synchronized time-triggered communication, global system time, and clock drift compensation. The FDCAN1 contains additional registers, specific to the time triggered feature. The CAN FD option can be used together with event-triggered and time-triggered CAN communication.

A 10 Kbyte message RAM implements filters, receive FIFOs, receive buffers, transmit event FIFOs, transmit buffers (and triggers for TTCAN). This message RAM is shared between the FDCAN modules.

The common clock calibration unit is optional. It can be used to generate a calibrated clock for each FDCAN from the HSi internal RC oscillator and the PLL, by evaluating CAN messages received by the FDCAN1.

The CAN subsystem I/O signals and pins are detailed, respectively, in Table 459 and Table 461.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fdcan_ker_ck</td>
<td>Digital input</td>
<td>CAN subsystem kernel clock input</td>
</tr>
<tr>
<td>fdcan_pclk</td>
<td>Digital input</td>
<td>CAN subsystem APB interface clock input</td>
</tr>
<tr>
<td>fdcan_cal_it</td>
<td>Digital output</td>
<td>FDCAN calibration interrupt</td>
</tr>
<tr>
<td>fdcan1_intr0_it</td>
<td>Digital output</td>
<td>FDCAN1 interrupt0</td>
</tr>
<tr>
<td>fdcan1_intr1_it</td>
<td>Digital output</td>
<td>FDCAN1 interrupt1</td>
</tr>
<tr>
<td>fdcan2_intr0_it</td>
<td>Digital output</td>
<td>FDCAN2 interrupt0</td>
</tr>
<tr>
<td>fdcan2_intr1_it</td>
<td>Digital output</td>
<td>FDCAN2 interrupt1</td>
</tr>
<tr>
<td>fdcan1.swt[0:3]</td>
<td>Digital input</td>
<td>Stop watch trigger input</td>
</tr>
<tr>
<td>fdcan1.ev[0:3]</td>
<td>Digital input</td>
<td>Event trigger input</td>
</tr>
<tr>
<td>fdcan1.ts[0:15]</td>
<td>Digital input</td>
<td>External timestamp vector</td>
</tr>
<tr>
<td>fdcan1_soc</td>
<td>Digital output</td>
<td>Start of cycle pulse</td>
</tr>
<tr>
<td>fdcan1_rtp</td>
<td>Digital output</td>
<td>Register time mark pulse</td>
</tr>
<tr>
<td>fdcan1_tmp</td>
<td>Digital output</td>
<td>Trigger time mark pulse</td>
</tr>
</tbody>
</table>
### Table 460. CAN triggers

<table>
<thead>
<tr>
<th>Type</th>
<th>ID</th>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal</td>
<td>SWT0</td>
<td>TIM2_TGO_CKTIM</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SWT1</td>
<td>TIM3_TGO_CKTIM</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SWT2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SWT3</td>
<td>TIM4_TGO_CKTIM</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>EVT0</td>
<td>TIM2_TGO_CKTIM</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>EVT1</td>
<td>TIM3_TGO_CKTIM</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>EVT2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>EVT3</td>
<td>TIM4_TGO_CKTIM</td>
<td>-</td>
</tr>
<tr>
<td>External</td>
<td>fdcan_tt_soc</td>
<td>-</td>
<td>TIM5_ITR6</td>
</tr>
<tr>
<td></td>
<td>fdcan_tt_tmp</td>
<td>-</td>
<td>TIM5_TI1_1</td>
</tr>
<tr>
<td></td>
<td>fdcan_tt_rtp</td>
<td>-</td>
<td>TIM5_TI1_2</td>
</tr>
</tbody>
</table>

### Table 461. CAN subsystem I/O pins

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDCAN1_RX</td>
<td>Digital</td>
<td>FDCAN1 receive pin</td>
</tr>
<tr>
<td>FDCAN1_TX</td>
<td>Digital</td>
<td>FDCAN1 transmit pin</td>
</tr>
<tr>
<td>FDCAN2_RX</td>
<td>Digital</td>
<td>FDCAN2 receive pin</td>
</tr>
<tr>
<td>FDCAN2_TX</td>
<td>Digital</td>
<td>FDCAN2 transmit pin</td>
</tr>
</tbody>
</table>
Figure 698. CAN subsystem

- Clock calibration unit (CCU)
- CAN core
- Control and Configuration registers
- Interrupts interface
- TX Handler
- RX Handler
- Message RAM interface
- RAM Controller / Arbiter
- Message RAM
  - Buffers
  - FIFOs
  - Filters
61.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B and ISO 11898-1: 2015, -4
- CAN FD with max. 64 data bytes supported
- TTCAN protocol level 1 and level 2 completely in hardware (FDCAN1 only)
- Event synchronized time-triggered communication supported (FDCAN1 only)
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two configurable receive FIFOs
- Separate signaling on reception of high priority messages
- Up to 64 dedicated receive buffers
- Up to 32 dedicated transmit buffers
- Configurable transmit FIFO / queue
- Configurable transmit event FIFO
- FDCAN modules share the same message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power-down support

61.3 FDCAN implementation

<table>
<thead>
<tr>
<th>Module</th>
<th>FDCAN1</th>
<th>FDCAN2</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTCAN</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 462. Main features
61.4 FDCAN functional description

Figure 699. FDCAN block diagram

Dual interrupt lines
The FDCAN peripheral provides two interrupt lines `fdcan_intr0_it` and `fdcan_intr1_it`. By programming EINT0 and EINT1 bits in FDCAN_IIE register, the interrupt lines can be enabled or disabled separately.

CAN core
The CAN core contains the protocol controller and receive/transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions and supports both 11-bit and 29-bit identifiers.

Sync
The sync block synchronizes signals from the APB clock domain to the CAN kernel clock domain and vice versa.
**Tx handler**

Controls the message transfer from the message RAM to the CAN core. A maximum of 32 Tx buffers can be configured for transmission. Tx buffers can be used as dedicated Tx buffers, as Tx FIFO, part of a Tx queue, or as a combination of them. A Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported.

On FDCAN1, the Tx handler also implements the frame synchronization entity (FSE) which controls time-triggered communication according to ISO11898-4. It synchronizes itself with the reference messages on the CAN bus, controls cycle time and global time, and handles transmissions according to the predefined message schedule, the system matrix. It also handles the time marks of the system matrix that are linked to the messages in the message RAM. Stop watch trigger, event trigger, and time mark interrupt are synchronization interfaces.

**Rx handler**

Controls the transfer of received messages from the CAN core to the external message RAM. The Rx handler supports two receive FIFOs, each of configurable size, and up to 64 dedicated Rx buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx buffer, in contrast to a receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

**APB Interface**

Connects the FDCAN to the APB bus.

**Message RAM Interface**

Connects the FDCAN access to an external 10 Kbytes message RAM through a RAM controller/arbiter.

### 61.4.1 Operating modes

**Software initialization**

Software initialization is started by setting INIT bit in FDCAN_CCCR register, either by software or by a hardware reset, or by going Bus_Off. While INIT bit in FDCAN_CCCR register is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output FDCAN_TX is recessive (high). The counters of the error management logic (EML) are unchanged. Setting INIT bit in FDCAN_CCCR does not change any configuration register. Clearing INIT bit in FDCAN_CCCR finishes the software initialization. Afterwards the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when both INIT bit in FDCAN_CCCR register and CCE bit in FDCAN_CCCR register are set.

CCE bit in FDCAN_CCCR register can only be set/cleared while INIT bit in FDCAN_CCCR is set. CCE bit in FDCAN_CCCR register is automatically cleared when INIT bit in FDCAN_CCCR is cleared.
The following registers are reset when CCE bit in FDCAN_CCCR register is set:
- FDCAN_HPMS - high priority message status
- FDCAN_RXF0S - Rx FIFO 0 status
- FDCAN_RXF1S - Rx FIFO 1 status
- FDCAN_TXFQS - Tx FIFO/queue status
- FDCAN_TXBRP - Tx buffer request pending
- FDCAN_TXBTO - Tx buffer transmission occurred
- FDCAN_TXBCF - Tx buffer cancellation finished
- FDCAN_TXEFS - Tx event FIFO status
- FDCAN_TTOST - TT (time trigger) operation status (FDCAN1 only)
- FDCAN_TTLGT - TT local and global time, only global time FDCAN_TTLGT.GT is reset (FDCAN1 only)
- FDCAN_TTCTC - TT cycle time and count (FDCAN1 only)
- FDCAN_TTCSM - TT cycle sync mark (FDCAN1 only)

The timeout counter value TOC bit in FDCAN_TOCV register is preset to the value configured by TOP bit in FDCAN_TOCC register when CCE bit in FDCAN_CCCR is set.

In addition the state machines of the Tx handler and Rx handler are held in idle state while CCE bit in FDCAN_CCCR is set.

The following registers can be written only when CCE bit in FDCAN_CCCR register is cleared:
- FDCAN_TXBAR - Tx buffer add request
- FDCAN_TXBCR - Tx buffer cancellation request

TEST bit in FDCAN_CCCR and MON bit in FDCAN_CCCR can only be set by software while both INIT bit and CCE bit in FDCAN_CCCR register are set. Both bits may be reset at any time. DAR bit in FDCAN_CCCR can only be set/cleared while both INIT bit in FDCAN_CCCR and CCE bit in FDCAN_CCCR are set.

**Normal operation**

The FDCAN1 default operating mode after hardware reset is event-driven CAN communication without time triggers (FDCAN_TTOCF.OM = 00). It is required that both INIT bit and CCE bit in FDCAN_CCCR register are set before the TT operation mode can be changed.

Once the FDCAN is initialized and INIT bit in FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into a dedicated Rx buffer or into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx buffers and/or a Tx FIFO or a Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

**CAN FD operation**

There are two variants in the CAN FD protocol, first the long frame mode (LFM) where the data field of a CAN frame may be longer that eight bytes. The second variant is the fast frame mode (FFM) where control field, data field, and CRC field of a CAN frame are
transmitted with a higher bitrate than the beginning and the end of the frame. Fast frame mode can be used in combination with long frame mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bitrate inside this CAN FD frame is switched. A CAN FD bitrate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for future expansion of the protocol.

If the FDCAN receives a frame with FDF recessive and res recessive, it will signal a protocol exception event by setting bit FDCAN_PS.R_PXE. When protocol exception handling is enabled (FDCAN_CCCR.PXH = 0), this causes the operation state to change from receiver (FDCAN_PSR.ACT = 10) to Integrating (FDCAN_PSR.ACT = 00) at the next sample point. In case protocol exception handling is disabled (FDCAN_CCCR.PXH = 1), the FDCAN will treat a recessive res bit as a form error and will respond with an error frame.

CAN FD operation is enabled by programming FDCAN_CCCR.FDOE. If FDCAN_CCCR.FDOE = 1, transmission and reception of CAN FD frames is enabled. Transmission and reception of classic CAN frames is always possible. Whether a CAN FD frame or a classic CAN frame is transmitted can be configured via bit FDF in the respective Tx buffer element. With FDCAN_CCCR.FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted, even if bit FDF of a Tx buffer element is set. FDCAN_CCCR.FDOE and FDCAN_CCCR.BRSE can only be changed while FDCAN_CCCR.INIT and FDCAN_CCCR.CCE are both set.

With FDCAN_CCCR.FDOE = 0, the setting of bits FDF and BRS is ignored and frames are transmitted in classic CAN format. With FDCAN_CCCR.FDOE = 1 and FDCAN_CCCR.BRSE = 0, only bit FDF of a Tx buffer element is evaluated. With FDCAN_CCCR.FDOE = 1 and FDCAN_CCCR.BRSE = 1, transmission of CAN FD frames with bitrate switching is enabled. All Tx buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bitrate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bitrate switching option for transmissions.

- During system startup all nodes are transmitting classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.

- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format.

- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to classic CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. DLC codes 0 to 8 have the same coding as in standard CAN, codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to Table 463.
In CAN FD fast frames, the bit timing will be switched inside the frame, after the BRS (bitrate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the bit timing and prescaler register FDCAN_NBTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the fast bit timing and prescaler register FDCAN_DBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bitrate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with a FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (tq), the bitrate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long frames and CAN FD fast frames, the value of the bit ESI (error status Indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case a FDCAN Tx buffer is configured for CAN FD transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives a FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx buffer or Rx FIFO. The remaining bytes are discarded.

**Transceiver delay compensation**

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. Without transceiver delay compensation, the bitrate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bitrates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit FDCAN_DBTP.TDC.

<table>
<thead>
<tr>
<th>DLC</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of data bytes</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>32</td>
<td>48</td>
<td>64</td>
</tr>
</tbody>
</table>
The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by FDCAN_TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq (minimum time quantum, that is one period of fdcan_to_ck clock).

FDCAN_PSR.TDCV shows the actual transmitter delay compensation value, it is cleared when FDCAN_CCCR.INIT is set and is updated at each transmission of an FD frame while FDCAN_DBTP.TDC is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCAN_TX to FDCAN_RX and the configured transmitter delay compensation offset FDCAN_TDCR.TDCO has to be less than six bit times in the data phase.
- The sum of the measured delay from FDCAN_TX to FDCAN_RX and the configured transmitter delay compensation offset FDCAN_TDCR.TDCO has to be less or equal 127 mtq. In case this sum exceeds 127 mtq, the maximum value (127 mtq) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, that stops checking received bits at the SSPs.

If transmitter delay compensation is enabled by programming FDCAN_DBTP.TDC = 1, the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is one mtq.

![Figure 700. Transceiver delay measurement](image URL)
programming FDCAN_TDCR.TDCF. This defines a minimum value for the SSP position. Dominant edges on FDCANx_RX, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least FDCAN_TDCR.TDCF and FDCAN_RX is low.

**Restricted operation mode**

In restricted operation mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters (FDCAN_ECR.REC, FDCAN_ECR.TEC) are frozen while error logging (FDCAN_ECR.CEL) is active. The software can set the FDCAN into restricted operation mode by setting bit FDCAN_CCCR.ASM. The bit can only be set by software when both FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set to 1. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx handler was not able to read data from the message RAM in time. To leave restricted operation mode, the software has to reset FDCAN_CCCR.ASM.

The restricted operation mode can be used in applications that adapt themselves to different CAN bitrates. In this case the application tests different bitrates and leaves the restricted operation mode after it has received a valid frame.

FDCAN_CCCR.ASM is also controlled by the clock calibration unit. When the clock calibration process is enabled, the restricted operation mode is entered and the FDCAN_CCCR.ASM bit is set. Once the calibration is completed, FDCAN_CCCR.ASM bit is cleared.

*Note:* The restricted operation mode must not be combined with the loop back mode (internal or external).

**Bus monitoring mode**

The FDCAN is set in bus monitoring mode by setting FDCAN_CCCR.MON bit or when error level S3 (FDCAN_TTOST.EL = 11) is entered. In bus monitoring mode (For more details please refer to ISO11898-1, 10.12 bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus, if the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In bus monitoring mode register FDCAN_TXBRP is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. Figure 701 shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in bus monitoring mode.
Disabled automatic retransmission (DAR) mode

According to the CAN Specification (see ISO11898-1, 6.3.3 recovery management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled.

To support time-triggered communication as described in ISO 11898-1: 2015, chapter 9.2, the automatic retransmission may be disabled via FDCAN_CCCR.DAR.

Frame transmission in disabled automatic retransmission (DAR) mode

In DAR mode all transmissions are automatically canceled after they started on the CAN bus. A Tx buffer Tx request pending bit FDCAN_TXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx set
- Arbitration loss or frame transmission disturbed:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx not set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx event FIFO element is written with event type ET = 10 (transmission in spite of cancellation).

Power down (Sleep mode)

The FDCAN can be set into power down mode controlled by clock stop request input via register FDCAN_CCCR.CSR. As long as the clock stop request is active, bit FDCAN_CCCR.CSR is read as 1.
When all pending transmission requests have completed, the FDCAN waits until bus idle state is detected. Then the FDCAN sets FDCAN_CCCR.INIT to 1 to prevent any further CAN transfers. Now the FDCAN acknowledges that it is ready for power down by setting FDCAN_CCCR.CSA to 1. In this state, before the clocks are switched off, further register accesses can be made. A write access to FDCAN_CCCR.INIT will have no effect. Now the module clock inputs may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting CC control register flag FDCAN_CCCR.CSR. The FDCAN will acknowledge this by resetting FDCAN_CCCR.CSA. Afterwards, the application can restart CAN communication by resetting bit FDCAN_CCCR.INIT.

Test modes

To enable write access to FDCAN test register (see Section 61.5.4), bit FDCAN_CCCR.TEST has to be set to 1, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming FDCAN_TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the FDCAN bit timing and it can drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from FDCAN_TEST.RX. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there may be a delay of several APB clock periods between writing to FDCAN_TEST.TX until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via FDCAN_TEST.RX.

Note: Test modes should be used for production tests or self test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External loop back mode

The FDCAN can be set in external loop back mode by programming FDCAN_TEST.LBCK to 1. In loop back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. Figure 702 (left side) shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in external loop back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop back mode. In this mode the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal loop back mode

Internal loop back mode is entered by programming bits FDCAN_TEST.LBCK and FDCAN_CCCR.MON to 1. This mode can be used for a “Hot Selftest”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, FDCAN_RX pin is disconnected from the FDCAN and FDCAN_TX pin is held recessive. Figure 702 (right side) shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of internal loop back mode.
Application watchdog (FDCAN1 only)

The application watchdog is served by reading register FDCAN_TTOST. When the application watchdog is not served in time, bit FDCAN_TTOST.AWE is set, all TTCAN communication is stopped, and the FDCAN1 is set into bus monitoring mode.

The TT application watchdog can be disabled by programming the application watchdog limit FDCAN_TTOCF.AWL to 0x00. The TT application watchdog should not be disabled in a TTCAN application program.

Timestamp generation

For timestamp generation the FDCAN supplies a 16-bit wraparound counter. A prescaler FDCAN_TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1 … 16). The counter is readable via FDCAN_TSCV.TCV. A write access to register FDCAN_TSCV resets the counter to 0. When the timestamp counter wraps around interrupt flag FDCAN_IR.TSW is set.

On start of frame reception/transmission the counter value is captured and stored into the timestamp section of a Rx buffer/Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming bit FDCAN_TSCC.TSS, a 16-bit timestamp can be used.

Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as down-counter and uses the same prescaler controlled by FDCAN_TSCC.TCP as the timestamp counter. The timeout counter is configured via register FDCAN_TOCC. The actual counter value can be read from FDCAN_TOCV.TOC. The timeout counter can only be started while FDCAN_CCCR.INIT = 0. It is stopped when FDCAN_CCCR.INIT = 1, e.g. when the FDCAN enters Bus_Off state.

The operation mode is selected by FDCAN_TOCC.TOS. When operating in Continuous mode, the counter starts when FDCAN_CCCR.INIT is reset. A write to FDCAN_TOCV
presets the counter to the value configured by FDCAN_TOCC.TOP and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by FDCAN_TOCC.TOP. Down-counting is started when the first FIFO element is stored. Writing to FDCAN_TOCV has no effect.

When the counter reaches 0, interrupt flag FDCAN_IR.TOO is set. In Continuous mode, the counter is immediately restarted at FDCAN_TOCC.TOP.

Note: The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore the point in time where the timeout counter is decremented may vary due to the synchronization/re-synchronization mechanism of the CAN core. If the baudrate switch feature in FDCAN is used, the timeout counter is clocked differently in arbitration and data fields.

61.4.2 Message RAM

The message RAM has a width of 32 bits. The FDCAN module can be configured to allocate up to 2560 words in the message RAM. It is not necessary to configure each of the sections listed in Figure 703, nor is there any restriction with respect to the sequence of the sections.

![Figure 703. Message RAM configuration](MS41464V2)

When the FDCAN addresses the message RAM it addresses 32-bit words, not single bytes. The configured start addresses are 32-bit word addresses, i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

Note: The FDCAN does not check for erroneous configuration of the message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.
Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to Rx buffers or to 1 of the two Rx FIFOs, as well as the Rx FIFO put and get Indices.

Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to Rx buffer, Rx FIFO 0 or Rx FIFO 1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from 0 to 128 elements for the 11-bit filter and from 0 to 64 for the 29-bit filter)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled/disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global filter configuration (FDCAN_GFC)
- Standard ID filter configuration (FDCAN_SIDFC)
- Extended ID filter configuration (FDCAN_XIDFC)
- Extended ID AND Mask (FDCAN_XIDAM)

Depending on the configuration of the filter element (SFEC / EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx buffer
- Store received frame in Rx buffer and generate pulse at filter event pin
- Reject received frame
- Set high priority message interrupt flag FDCAN_IR.HPM
- Set high priority message interrupt flag FDCAN_IR.HPM and store received frame in FIFO 0 or FIFO 1
- Set high priority message interrupt flag FDCAN_IR.HPM and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx buffer or Rx FIFO has been found, the message handler starts writing the received message data in portions of 32 bit to the
matching Rx buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact:

- **Rx buffer**
  New data flag of matching Rx buffer is not set, but Rx buffer (partly) overwritten with received data. For error type see FDCAN_PSR.LEC and FDCAN_PSR.DLEC.

- **Rx FIFO**
  Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see FDCAN_PSR.LEC and FDCAN_PSR.DLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in **Rx FIFO overwrite mode** have to be considered.

**Note:** When an accepted message is written to one of the two Rx FIFOs, or into an Rx buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

**Range filter**

The filter matches for all received frames with message IDs in the range defined by SF1ID / SF2ID and EF1ID / EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- **EFT = 00:** The message ID of received frames is AND-ed with the extended ID AND Mask (FDCAN_XIDAM) before the range filter is applied
- **EFT = 11:** The extended ID AND Mask (FDCAN_XIDAM) is not used for range filtering

**Filter for dedicated IDs**

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SF1ID = SF2ID and EF1ID = EF2ID.

**Classic bit mask filter**

Classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SF1ID / EF1ID is used as message ID filter, while SF2ID / EF2ID is used as filter mask.

A 0 bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received message ID where the corresponding mask bits are one are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.
Standard message ID filtering

*Figure 704* shows the flow for standard message ID (11-bit Identifier) filtering. The standard message ID filter element is described in *Section 61.4.21.*

![Flowchart for Standard Message ID Filtering](image)

Controlled by the global filter configuration (FDCAN_GFC) and the standard ID filter configuration (FDCAN_SIDFC) message ID, remote transmission request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Extended message ID filtering

*Figure 705* shows the flow for extended message ID (29-bit Identifier) filtering. The extended message ID filter element is described in *Section 61.4.22.*
Controlled by the global filter configuration FDCAN_GFC and the extended ID filter configuration FDCAN_XIDFC message ID, remote transmission request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The extended ID AND Mask (FDCAN_XIDAM) is AND-ed with the received identifier before the filter list is executed.

**Rx FIFOs**

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers FDCAN_RXF0C and FDCAN_RXF1C.
Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see Acceptance filter. The Rx buffer and FIFO element is described in Section 61.4.18.

When an Rx FIFO full condition is signaled by FDCAN_IR.RFnF, no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded and interrupt flag FDCAN_IR.RFnL is set.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by FDCAN_RXFnC.FnWM, interrupt flag FDCAN_IR.RFnW is set.

When reading from an Rx FIFO, Rx FIFO get index RXFnS[FnGI] + FIFO element size has to be added to the corresponding Rx FIFO start address RXFnC[FnSA].

**Rx FIFO blocking mode**

The Rx FIFO blocking mode is configured by RXFnC.FnOM = 0. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled by RXFnS.FnF = 1. In addition interrupt flag FDCAN_IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signaled by RXFnS.RFnL = 1. In addition interrupt flag FDCAN_IR.RFnL is set.

**Rx FIFO overwrite mode**

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = 1.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signaled by RXFnS.FnF = 1, the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is that it can happen that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. Figure 707 shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in elements 1 and 2 are lost.

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO acknowledge index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = 0).

**Dedicated Rx buffers**

The FDCAN supports up to 64 dedicated Rx buffers. The start address of the dedicated Rx buffer section is configured via FDCAN_RXBC.RBSA.
For each Rx buffer a standard or extended message ID filter element with SFEC / EFEC=111 and SFID2 / EFID2[10:9] = 00 has to be configured (see Section 61.4.21 and Section 61.4.22).

After a received message has been accepted by a filter element, the message is stored into the Rx buffer in the message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag FDCAN_IR.DRX (message stored in dedicated Rx buffer) in the interrupt register is set.

After the last word of a matching received message has been written to the message RAM, the respective New data flag in register NDAT1,2 is set. As long as the New data flag is set, the respective Rx buffer is locked against updates from received matching frames. The New data flags have to be reset by the Host by writing a 1 to the respective bit position.

While an Rx buffer New data flag is set, a message ID filter element referencing this specific Rx buffer will not match, causing the acceptance filtering to continue. The following message ID filter elements may cause the received message to be stored into another Rx buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

**Rx buffer handling**

- Reset interrupt flag FDCAN_IR.DRX
- Read New data registers
- Read messages from message RAM
- Reset New data flags of processed messages

**Filtering for Debug messages**

Filtering for debug messages is done by configuring one standard/extended message ID filter element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC/EFEC has to be programmed to 111. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning. While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New data flag nor FDCAN_IR.DRX are set. The reception of debug messages can be monitored via FDCAN_RXF1S.DMS.

### Table 464. Example of filter configuration for Rx buffers

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID message 1</td>
<td>00</td>
<td>00 0000</td>
</tr>
<tr>
<td>1</td>
<td>ID message 2</td>
<td>00</td>
<td>00 0001</td>
</tr>
<tr>
<td>2</td>
<td>ID message 3</td>
<td>00</td>
<td>00 0010</td>
</tr>
</tbody>
</table>

After the last word of a matching received message has been written to the message RAM, the respective New data flag in register NDAT1,2 is set. As long as the New data flag is set, the respective Rx buffer is locked against updates from received matching frames. The New data flags have to be reset by the Host by writing a 1 to the respective bit position.
Tx handling

The Tx handler handles transmission requests for the dedicated Tx buffers, the Tx FIFO, and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get Indices, and the Tx event FIFO. Up to 32 Tx buffers can be set up for message transmission (see Dedicated Tx buffers). Depending on the configuration of the element size (FDCAN_RXESC), between two and sixteen 32-bit words (Rn = 3 ... 17) are used for storage of a CAN message data field.

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register FDCAN_TXBRP is updated, or when a transmission has been started.

Transmit pause

This feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If, as an example, CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID debug message A</td>
<td>01</td>
<td>11 1101</td>
</tr>
<tr>
<td>1</td>
<td>ID debug message B</td>
<td>10</td>
<td>11 1110</td>
</tr>
<tr>
<td>2</td>
<td>ID debug message C</td>
<td>11</td>
<td>11 1111</td>
</tr>
</tbody>
</table>

Table 465. Example of filter configuration for Debug messages

<table>
<thead>
<tr>
<th>FDCAN_CCCR</th>
<th>Tx buffer element</th>
<th>Frame transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRSE</td>
<td>FDOE</td>
<td>FDF</td>
</tr>
<tr>
<td>Ignored</td>
<td>0</td>
<td>Ignored</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 466. Possible configurations for frame transmission

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.
received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by TXP bit in FDCAN_CCCR register. If the bit is set, the FDCAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower priority identifiers. Default is disabled (FDCAN_CCCR.TXP = 0).

This feature loosens up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

**Dedicated Tx buffers**

Dedicated Tx buffers are intended for message transmission under complete control of the CPU. Each dedicated Tx buffer is configured with a specific message ID. In case that multiple Tx buffers are configured with the same message ID, the Tx buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an add request via FDCAN_TXBAR.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx queue and externally with messages on the CAN bus, and are sent out according to their message ID.

A dedicated Tx buffer allocates four 32-bit words in the message RAM. Therefore the start address of a dedicated Tx buffer in the message RAM is calculated by adding four times the transmit buffer index (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA.

### Table 467. Tx buffer/FIFO - queue element size

<table>
<thead>
<tr>
<th>FDCAN_TXESC.TBDS[2;0]</th>
<th>Data field (bytes)</th>
<th>Element size (RAM words)</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>001</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>010</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>011</td>
<td>20</td>
<td>7</td>
</tr>
<tr>
<td>100</td>
<td>24</td>
<td>8</td>
</tr>
<tr>
<td>101</td>
<td>32</td>
<td>10</td>
</tr>
<tr>
<td>110</td>
<td>48</td>
<td>14</td>
</tr>
<tr>
<td>111</td>
<td>64</td>
<td>18</td>
</tr>
</tbody>
</table>

**Tx FIFO**

Tx FIFO operation is configured by programming FDCAN_TXBC.TFQM to 0. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index FDCAN_TXFQS.TFGI. After each transmission the get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO free level FDCAN_TXFQS.TFFL as difference between get and put index. It indicates the number of available (free) Tx FIFO elements.
New transmit messages have to be written to the Tx FIFO starting with the Tx buffer referenced by the put index FDCAN_TXFQS.TFQPI. An add request increments the put index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO Full (FDCAN_TXFQS.TFQF = 1) is signaled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a 1 to the FDCAN_TXBAR bit related to the Tx buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the put index. The transmissions are then requested via FDCAN_TXBAR. The put index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx buffers as indicated by the Tx FIFO free level.

When a transmission request for the Tx buffer referenced by the get index is canceled, the get index is incremented to the next Tx buffer with pending transmission request and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the get index and the FIFO free level remain unchanged.

A Tx FIFO element allocates four 32-bit words in the message RAM. Therefore the start address of the next available (free) Tx FIFO buffer is calculated by adding four times the put index FDCAN_TXFQS.TFQPI (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA.

**Tx queue**

Tx queue operation is configured by programming FDCAN_TXBC.TFQM to 1. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority). In case that multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx buffer referenced by the put index FDCAN_TXFQS.TFQPI. An add request cyclically increments the put index to the next free Tx buffer. In case that the Tx queue is full (FDCAN_TXFQS.TFQF = 1), the put index is not valid and no further message should be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

The application may use register FDCAN_TXBRP instead of the put index and may place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates four 32-bit words in the message RAM. Therefore the start address of the next available (free) Tx queue buffer is calculated by adding four times the Tx queue put index FDCAN_TXFQS.TFQPI (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA.

**Mixed dedicated Tx buffers / Tx FIFO**

In this case the Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx FIFO. The number of dedicated Tx buffers is configured by FDCAN_TXBC.NDTB. The number of Tx buffers assigned to the Tx FIFO is configured by FDCAN_TXBC.TFQS. In case, FDCAN_TXBC.TFQS is programmed to 0, only dedicated Tx buffers are used.
**Figure 706. Example of mixed configuration dedicated Tx buffers / Tx FIFO**

<table>
<thead>
<tr>
<th>Buffer index</th>
<th>Dedicated Tx Buffers</th>
<th>Tx FIFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID3</td>
<td>ID4</td>
</tr>
<tr>
<td>1</td>
<td>ID15</td>
<td>ID2</td>
</tr>
<tr>
<td>2</td>
<td>ID8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ID24</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tx prioritization:**
- Scan dedicated Tx buffers and oldest pending Tx FIFO buffer (referenced by FDCAN_TXFS.TFGI)
- Buffer with lowest message ID gets highest priority and is transmitted next

**Mixed dedicated Tx buffers / Tx queue**

In this case the Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx queue. The number of dedicated Tx buffers is configured by FDCAN_TXBC.NDTB. The number of Tx queue buffers is configured by FDCAN_TXBC.TFQS. If FDCAN_TXBC.TFQS is programmed to 0, only dedicated Tx buffers are used.

**Figure 707. Example of mixed configuration dedicated Tx buffers / Tx queue**

<table>
<thead>
<tr>
<th>Buffer index</th>
<th>Dedicated Tx Buffers</th>
<th>Tx Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID3</td>
<td>ID4</td>
</tr>
<tr>
<td>1</td>
<td>ID15</td>
<td>ID2</td>
</tr>
<tr>
<td>2</td>
<td>ID8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ID24</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tx priority setting:**
- Scan all Tx buffers with activated transmission request
- Tx buffer with lowest message ID gets highest priority and is transmitted next
Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a dedicated Tx buffer or a Tx queue buffer the Host has to write a 1 to the corresponding bit position (= number of Tx buffer) of register FDCAN_TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register FDCAN_TXBCF to 1.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding FDCAN_TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding FDCAN_TXBTO and FDCAN_TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding FDCAN_TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is pending in this node. This may enable another node to transmit a message that may have a priority lower than that of the second message in this node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO can be configured to a maximum of 32 elements. The Tx event FIFO element is described in Tx FIFO. Depending on the configuration of the element size (FDCAN_TXESC), between two and sixteen 32-bit words (Tn = 3 ..17) are used for storage of a CAN message data field.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling i.e. a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by FDCAN_IR.TEFF, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and interrupt flag FDCAN_IR.TEFL is set.

To avoid a Tx event FIFO overflow, the Tx event FIFO watermark can be used. When the Tx event FIFO fill level reaches the Tx event FIFO watermark configured by FDCAN_TXEFC.EFWM, interrupt flag FDCAN_IR.TEFW is set.

When reading from the Tx event FIFO, two times the Tx event FIFO get index FDCAN_TXEFS.EFGI has to be added to the Tx event FIFO start address FDCAN_TXEFC.EFSA.

61.4.3 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index, see FDCAN Rx FIFO 0 acknowledge register.
Writing to the FIFO acknowledge index will set the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

1. When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO acknowledge index.
2. When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value: index of the last element read), to update the FIFO get index.

Due to the fact that the CPU has free access to the FDCAN message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case the FIFO acknowledge index should not be written because this would set the get index to a wrong position and also alters the FIFO fill level. In this case some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO acknowledge index. The FDCAN does not check for erroneous values.

61.4.4 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

As shown in Figure 708, its operation may be explained simply by splitting the bit time in three segments, as follows:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, that has a fixed length of one time quantum (1 x tq).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard, its duration is programmable between 1 and 16 time quanta, but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between one and eight time quanta, but may also be automatically shortened to compensate for negative phase drifts.
The baudrate is the inverse of bit time (baudrate = 1 / bit time), which, in turn, is the sum of three components. Figure 708 indicates that bit time = \( t_{\text{SyncSeg}} + t_{\text{BS1}} + t_{\text{BS2}} \), where:

- for the nominal bit time
  - \( t_q = (\text{FDCAN\_NBTP\_NBRP}[8:0] + 1) \times t_{\text{fdcan\_tq\_ck}} \)
  - \( t_{\text{SyncSeg}} = 1 \times t_q \)
  - \( t_{\text{BS1}} = t_q \times (\text{FDCAN\_NBTP\_NTSEG1}[7:0] + 1) \)
  - \( t_{\text{BS2}} = t_q \times (\text{FDCAN\_NBTP\_NTSEG2}[6:0] + 1) \)

- for the data bit time
  - \( t_q = (\text{FDCAN\_DBTP\_DBRP}[4:0] + 1) \times t_{\text{fdcan\_tq\_ck}} \)
  - \( t_{\text{SyncSeg}} = 1 \times t_q \)
  - \( t_{\text{BS1}} = t_q \times (\text{FDCAN\_DBTP\_DTSEG1}[4:0] + 1) \)
  - \( t_{\text{BS2}} = t_q \times (\text{FDCAN\_DBTP\_DTSEG2}[3:0] + 1) \)

The (re)synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the bit timing register is only possible while the device is in Standby mode. Registers FDCAN\_DBTP and FDCAN\_NBTP (dedicated, respectively, to data and nominal bit timing) are only accessible when FDCAN\_CCR\_CCE and FDCAN\_CCR\_INIT are set.

Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.

61.4.5 Clock calibration on CAN

After device reset the clock calibration unit (CCU) does not provide a valid clock signal to the FDCAN(s). The CCU has to be initialized via FDCAN\_CCFG register. The FDCAN\_CCFG register can be written only when FDCAN1 has both FDCAN\_CCR\_CCE and FDCAN\_CCR\_INIT bits set. In consequence the CCU and the FDCAN\_CCR initialization needs to be completed before any FDCAN module can operate.

Clock calibration is bypassed when FDCAN\_CCFG\_BCC = 1 (see Figure 709).
Operating conditions
The clock calibration on CAN unit is designed to operate under the following conditions:
- a CAN kernel clock frequency \( f_{\text{can.ker.ck}} \) up to 80 MHz
- FDCAN bitrates:
  - Nominal bitrate: up to 1 Mbit/s
  - Data bitrate: between nominal bitrate and 8 Mbit/s

The clock calibration on FDCAN unit generates a calibrated time quanta clock \( f_{\text{can.tq.ck}} \) in the range from 0.5 to 25 MHz.

Note: The FDCAN requires that the CAN time quanta clock is always below or equal to the APB clock \( (f_{\text{can.tq.ck}} < f_{\text{can.pclk}}) \). This has to be considered when the clock calibration on CAN unit is bypassed \((\text{FDCAN.CCFG.BCC} = 1)\).

Calibration accuracy
The calibration accuracy in state Precision_Calibrated depends upon the factors listed below.
- Dynamic clock tolerance at the CAN kernel clock input \( f_{\text{can.ker.ck}} \)
- Measurement error. For each bit sequence used for calibration measurement, there is a maximum error of one \( f_{\text{can.pclk}} \) period. The number of bits used for measurement of the bit time is 32 or 64-bit, depending on configuration of FDCAN_CCFG.CFL.
- Tolerable error in calibration mechanism

The distance between two calibration messages has to be chosen to fit the clock tolerance requirements of the FDCAN1 module.

Note: Dynamic clock tolerance is the clock frequency variation between two calibration messages e.g. caused by change of temperature or operating voltage.

Functional description
Calibration of the time quanta clock \( f_{\text{can.tq.ck}} \) via CAN messages is performed by adapting a clock divider that generates the CAN protocol time quantum \( tq \) from the clock \( f_{\text{can.ker.ck}} \).

1. First step: basic calibration
   The minimum distance between two consecutive falling edges from recessive to dominant is measured, this time to be assumed two CAN bit times, counted in PLL clock periods. The clock divider (FDCAN_CCFG.CDIV) is updated each time a new
measurement finds a smaller distance between edges. Basic calibration is achieved when the CAN protocol controller detects a valid CAN message.

2. Second step: Precision calibration
The calibration state machine measures the length of a longer bit sequence inside a CAN frame by counting the number of fdcan_ker_ck periods. The length of this bit sequence can be configured to 32 or 64 bits via FDCAN_CCFG.CLF. For a calibration field length of 32/64 bit a calibration message with at least 2/4 byte data field is required. Precision calibration is based on the new clock divider value calculated from the measurement of the longer bit sequence.

Figure 710. FSM calibration

<table>
<thead>
<tr>
<th>Event</th>
<th>Diagram State</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0: HW reset</td>
<td>Not_Calibrated</td>
<td>HW reset set, CALS=00</td>
</tr>
<tr>
<td>T1: Check for minimum</td>
<td>Not_Calibrated</td>
<td>CALS=00</td>
</tr>
<tr>
<td>T2: Message received</td>
<td>Basic_Calibrated</td>
<td>CALS=01</td>
</tr>
<tr>
<td>T3: Evaluate messages</td>
<td>Precision_Calibrated</td>
<td>CALS=00</td>
</tr>
<tr>
<td>T4: Quartz message received</td>
<td>Basic_Calibrated</td>
<td>CALS=01</td>
</tr>
<tr>
<td>T5: T4: Quartz message received</td>
<td>Basic_Calibrated</td>
<td>CALS=01</td>
</tr>
<tr>
<td>T6: Configuration change enabled</td>
<td>Precision_Calibrated</td>
<td>CALS=00</td>
</tr>
<tr>
<td>T7: Watchdog event: no quartz message received</td>
<td>Precision_Calibrated</td>
<td>CALS=00</td>
</tr>
<tr>
<td>T8: Watchdog event: no quartz message received or configuration change enabled</td>
<td>Precision_Calibrated</td>
<td>CALS=00</td>
</tr>
</tbody>
</table>

A change in the calibration state sets interrupt flag FDCAN_CCU_IR.CSC, if enabled by the interrupt enable FDCAN_CCU_IE.CSCE. It remains set until cleared by writing 1 in FDCAN_CCU_IR.CSC.

Until precision calibration is achieved, FDCANs operate in a restricted mode (no frame transmission, no error or overload flag transmission, no error counting). In case calibration of the fdcan_ker_clk is done by software by evaluating the calibration status from register FDCAN_CCU_CSTAT, FDCANs have to be set to restricted operation mode (FDCAN_CCCR.ASM = 1) until the calibration on CAN unit is in state Precision_Calibrated (see Application).

Precision calibration may be performed only on valid CAN frames transmitted by a node with a stable, quartz-controlled clock. Calibration frames are detected by the FDCAN1 acceptance filtering A filter element and a Rx buffer have to be configured in the FDCAN1 to identify and store calibration messages. After reception of a calibration message the Rx buffer new data flag has to be reset to enable signaling of the next calibration message.
In case there is only one CAN transmitter with a quartz clock in the network, this node has to transmit its first message after startup with at least one 1010 binary sequence in the data field or in the identifier. This assures that the non-quartz nodes can enter state Basic_Calibrated and then acknowledge the quartz node messages.

Precision calibration must be repeated in predefined maximum intervals supervised by the calibration watchdog.

**Note:** When the clock calibration on CAN unit transits from state Precision_Calibrated back to Basic_Calibrated, the calibration OK signal is deasserted, the FDCAN1 complete ongoing transmissions, and then enter restricted operation (no frame transmission, no error or overload flag transmission, no error counting).

### Configuration

The clock calibration on CAN unit is configured via register FDCAN_CCFG, i.e. when FDCAN1 has FDCAN_CCCR.CCE and FDCAN_CCCR.INIT bits set.

For basic calibration the minimum number of oscillator periods between two consecutive falling edges at pin FDCAN1_RX is measured. The number of clock periods depends on the clock frequency applied at input fdcan_ker_ck. In case the measured number of clock periods is below the minimum configured by FDCAN_CCFG.OCPM (as an example, because of a glitch on FDCAN1_RX) the value is discarded and measurement continues.

It is recommended to configure FDCAN_CCFG.OCPM slightly below two CAN bit times:

\[
\text{FDCAN_CCFG.OCPM} < \left(\frac{2 \times \text{CAN bit time}}{\text{fdcan_ker_ck period}}\right) / 32
\]

The length of the bit field used for precision calibration can be configured to 32 or 64 bits via FDCAN_CCFG.CFL. The number of bits used for precision calibration has an impact on calibration accuracy and the maximum distance between two calibration messages.

The number of time quanta per bit time configured by FDCAN_CCFG.TQBT is used together with the measured number of oscillator clock periods FDCAN_CCU_CSTAT.OCPC to define the number of oscillator clocks per bit time.

When the clock calibration is bypassed by configuring FDCAN_CCFG.BCC = 1, the internal clock divider has to be configured via FDCAN_CCFG.CDIV to fulfill the condition fdcan_to_ck < fdcan_pclk.

**Note:** When clock calibration on CAN is active (FDCAN_CCFG.BCC = 0), the baudrate prescalers of FDCAN modules have to be configured to inactive.

### Status signaling

The status of the clock calibration on CAN unit can be monitored by reading register FDCAN_CCU_CSTAT. When in state Precision_Calibrated the oscillator clock period counter FDCAN_CCU_CSTAT.OCPC signals the number of oscillator clock periods in the calibration field while FDCAN_CCU_CSTAT.TQC signals the number of time quanta in the calibration field.

The calibration state is monitored by FDCAN_CCU_CSTAT.CALS. A change in the calibration state sets interrupt flag FDCAN_CCU_IR.CSC, if enabled by the interrupt enable FDCAN_CCU_IE.CSCE it remains set until cleared by writing 1 in FDCAN_CCU_IE.CSC.

A calibration watchdog event also sets interrupt flag FDCAN_CCU_IR.CWE, if enabled by FDCAN_CCU_IE.CWEE (set to high) it remains active until reset by FDCAN_CCU_IE.CWE.
61.4.6 Application

Clock calibration bypassed

The CCU internal clock divider is configured for division by one (FDCAN_CCFG.CDIV = 0x0000). In this operation mode the input clock fdcan_ker_ck is directly routed to the clock output fdcan_tq_ck. In this case fdcan_tq_ck is independent from the configuration and status of FDCAN1 and FDCAN2 connected to the CCU. CAN FD operation is possible with a fdcan_ker_ck up to 80 MHz.

Software calibration

The clock calibration on CAN unit also supports software calibration of fdcan_ker_ck by trimming of an on-chip oscillator. For calculation of the trimming values the Host has to read the CCU state from FDCAN_CCU_CSTAT. In this operation mode the CCU output signal cu_cok is fixed to 1, the clock from fdcan_ker_ck is routed to output fdcan_tq_ck (FDCAN_CCFG.BCC = 1).

The input clock fdcan_ker_ck must be in the range between 80MHz and 100 MHz. The clock divider of CCU has to be configured via FDCAN_CCFG.CDIV to bring fdcan_tq_ck to a valid range. All other configuration parameters have to be set via FDCAN_CCFG. For correct operation of FDCAN1 and FDCAN2, the APB clock fdcan_pclk needs to be equal or higher than the time quanta clock (fdcan_tq_ck). CAN FD operation is not possible.

For startup FDCAN modules have to be both configured for restricted operation (FDCAN_CCCR.ASM = 1) before FDCAN_CCCR.INIT is reset. The input clock fdcan_ker_ck has to be adjusted until the clock calibration on CAN unit has reached state Precision_Calibrated. Now the software can reset FDCAN_CCCR.ASM and the CANFD1 and CANFD2 can start normal operation.

During operation the software has to check regularly whether the clock calibration on CAN unit is still in state Precision_Calibrated. In case the clock calibration on CAN unit has left state Precision_Calibrated due to drift of fdcan_ker_ck, FDCAN modules have to be set into restricted operation mode by programming FDCAN_CCCR.INIT, FDCAN_CCCR.CCE, and FDCAN_CCCR.ASM to 1. After fdcan_ker_ck has been adjusted successfully (clock calibration on CAN unit is in state Precision_Calibrated), FDCAN modules can resume normal operation.

Note: Trimming accuracy must be to sufficient to meet the CAN clock tolerance requirements for the configured bitrate.

Clock calibration active

This operation mode is entered by resetting FDCAN_CCFG.BCC to 0. In this operation mode the CCU output signals cu_cok and fdcan_ker_ck are controlled by the CCU. When the CCU is not in state Precision_Calibrated cu_cok is 0.

Generation of CCU output signals fdcan_tq_ck and cu_cok depends on the state of the FDCAN1. Input clock fdcan_ker_ck must be below 80 MHz. Configuration of the CCU and FDCAN1 is required. CAN FD operation is not possible.

In case FDCAN1 turns to Bus_Off or when its INIT bit is set by Host command (FDCAN_CCCR.INIT = 1), the CCU enters state Not_Calibrated and output cu_cok is reset to 0. CANFD1 and CANFD2 enter restricted operation mode.

Note: This is the default operation mode after reset in case the reset value of FDCAN_CCFG.BCC is configured to 0.
61.4.7 TTCAN operations (FDCAN1 only)

Reference message

A reference message is a data frame characterized by a specific CAN identifier. It is received and accepted by all nodes except the time master (sender of the reference message).

For level 1 the data length must be at least one; for level 0, 2 the data length must be at least four; otherwise, the message is not accepted as reference message. The reference message may be extended by other data up to the sum of eight CAN data bytes. All bits of the identifier except the three LSBs characterize the message as a reference message. The last three bits specify the priorities of up to eight potential time masters. Reserved bits are transmitted as logical 0 and are ignored by the receivers. The reference message is configured via register FDCAN_TTRMC.

The time master transmits the reference message. If the reference message is disturbed by an error, it is retransmitted immediately. In case of a retransmission, the transmitted Master_Ref_Mark is updated. The reference message is sent periodically, but is allowed to stop the periodic transmission (Next_is_Gap bit) and to initiate transmission event-synchronized at the start of the next basic cycle by the current time master or by one of the other potential time masters.

The node transmitting the reference message is the current time master. The time master is allowed to transmit other messages. If the current time master fails, its function is replicated by the potential time master with the highest priority. Nodes that are neither time master nor potential time master are time-receiving nodes.

Level 1

Level 1 operation is configured via FDCAN_TTOCF.OM = 01 and FDCAN_TTOCF.GEN. external clock synchronization is not available in level 1. The information related to the reference message is stored in the first data byte as shown in Table 468. Cycle_Count is optional.

<table>
<thead>
<tr>
<th>Bits</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>First byte</td>
<td>Next_is_Gap</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td>Cycle_Count[5;0]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Level 2

Level 2 operation is configured via FDCAN_TTOCF.OM = 10 and FDCAN_TTOCF.GEN. The information related to the reference message is stored in the first four data bytes as shown in Table 469. Cycle_Count and the lower four bits of FDCAN_NTU_Res are optional. The TTCAN does not evaluate NTU_Res[3;0] from received reference messages, it always transmits these bits as 0.
Level 0

Level 0 operation is configured via FDCAN_TTOCF.OM = 11. External event-synchronized time-triggered operation is not available in level 0. The information related to the reference message is stored in the first four data bytes as shown in the table below. In level 0 Next_is_Gap is always 0. Cycle_Count and the lower four bits of NTU_Res are optional. The TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as 0.

<table>
<thead>
<tr>
<th>Bits</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>First byte</td>
<td>Next_is_Gap</td>
<td>Reserved</td>
<td>Cycle_Count[5:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Third byte</td>
<td></td>
<td>Master_Ref_Mark[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fourth byte</td>
<td></td>
<td>Master_Ref_Mark[15:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

61.4.8 TTCAN configuration

TTCAN timing

The network time unit (NTU) is the unit in which all times are measured. The NTU is a constant of the whole network and is defined as a priority by the network system designer. In TTCAN level 1 the NTU is the nominal CAN bit time. In TTCAN level 0 and level 2 the NTU is a fraction of the physical second.

The NTU is the time base for the local time. The integer part of the local time (16-bit value) is incremented once each NTU. Cycle time and global time are both derived from local time. The fractional part (3-bit value) of local time, cycle time, and global time is not readable.

In TTCAN level 0 and level 2 the length of the NTU is defined by the time unit Ratio TUR. The TUR is in general a non-integer number, given by TUR = FDCAN_TURNA.NAV / FDCAN_TURCF.DC. The NTU length is given by NTU = CAN clock period $\times$ TUR.
The TUR numerator configuration NC is an 18-bit number, FDCAN_TURCF[NCL[15:0]] can be programmed in the range 0x0000–0xFFFF. FDCAN_TURCF[NCH[17:16]] is hard wired to 0b01. When 0xnnnn is written to FDCAN_TURCF[NCL[15:0]], FDCAN_TURNA.NAV starts with the value 0x10000 + 0x0nnnn = 0x1nnnn. The TUR denominator configuration FDCAN_TURCF.DC is a 14-bit number. FDCAN_TURCF.DC may be programmed in the range 0x0001 - 0x3FFF (0x0000 is an illegal value).

In level 1, NC must be equal or higher than 4 x FDCAN_TURCF.DC. In level 0 and level 2 NC must be equal or higher than 8 x FDCAN_TURCF.DC to get the 3-bit resolution for the internal fractional part of the NTU.

A hardware reset presets FDCAN_TURCF.DC to 0x1000 and FDCAN_TURCF.NCL to 0x10000, resulting in an NTU consisting of sixteen CAN clock periods. Local time and application watchdog are not started before either the FDCAN_CCCR.INIT is reset, or FDCAN_TURCF.ELT is set. FDCAN_TURCF.ELT may not be set before the NTU is configured. Setting FDCAN_TURCF.ELT to 1 also locks the write access to register FDCAN_TURCF.

At startup FDCAN_TURNA.NAV is updated from NC (= FDCAN_TURCF.NCL + 0x10000) when FDCAN_TURCF.ELT is set. In TTCAN level 1 there is no drift compensation. FDCAN_TURNA.NAV does not change during operation, it is always equal to NC.

In TTCAN level 0 and level 2 there are two possibilities for FDCAN_TURNA.NAV to change. When operating as time slave or backup time master, and when FDCAN_TTOCF.ECC is set, FDCAN_TURNA.NAV is updated automatically to the value calculated from the monitored global time speed, as long as the TTCAN is in synchronization state In_Schedule or In_Gap. When it loses synchronization, it returns to NC. When operating as the actual time master, and when FDCAN_TTOCF.EECS is set, the Host may update FDCAN_TURCF.NCL. When the Host sets FDCAN_TTOCN.ECS, FDCAN_TURNA.NAV will be updated from the new value of NC at the next reference message. The status flag FDCAN_TTOST.WECS as is set when FDCAN_TTOCN.ECS is set and is cleared when FDCAN_TURNA.NAV is updated. FDCAN_TURCF.NCL is write locked while FDCAN_TTOST.WECS is set.

In TTCAN level 0 and level 2 the clock calibration process adapts FDCAN_TURNA.NAV in the range of the synchronization deviation limit SDL of NC ± 2(FDCAN_TTOCF.LDSDL + 5). FDCAN_TURCF.NCL should be programmed to the largest applicable numerical value in order to achieve the best accuracy in the calculation of FDCAN_TURNA.NAV.

The synchronization deviation SD is the difference between NC and FDCAN_TURNA.NAV (SD = | NC - FDCAN_TURNA.NAV |). It is limited by the synchronization deviation limit SDL, which is configured by its dual logarithm FDCAN_TTOCF.LDSDL (SDL = 2(FDCAN_TTOCF.LDSDL + 5)) and should not exceed the clock tolerance given by the CAN bit timing configuration. SD is calculated at each new basic cycle. When the calculated TURNA[NAV deviates by more than SDL from NC, or if the Disc_Bit in the reference message is set, the drift compensation is suspended and FDCAN_TTIR.GTE is set and FDCAN_TTOSC.QCS is reset, or in case of the Disc_Bit = 1, FDCAN_TTIR.GTD is set.

### Table 471. TUR configuration example

<table>
<thead>
<tr>
<th>TUR</th>
<th>8</th>
<th>10</th>
<th>24</th>
<th>50</th>
<th>510</th>
<th>125000</th>
<th>32.5</th>
<th>100/12</th>
<th>529/17</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>0x1FFF8</td>
<td>0x1FFFFE</td>
<td>0x1FFF8</td>
<td>0x1FFFE</td>
<td>0x1FEEA</td>
<td>0x1FFE0</td>
<td>0x1E848</td>
<td>0x1EFE0</td>
<td>0x19000</td>
</tr>
</tbody>
</table>
FDCAN_TTOCN.ECS schedules NC for activation by the next reference message.
FDCAN_TTOCN.SGT schedules FDCAN_TTGTP.TP for activation by the next reference message. Setting FDCAN_TTOCN.ECS and FDCAN_TTOCN.SGT requires
FDCAN_TTOCF.EECS to be set (external clock synchronization enabled) while the FDCAN is actual time master.

The TTCAN module provides an application watchdog to verify the function of the application program. The Host has to serve this watchdog regularly, else all CAN bus activity is stopped. The application watchdog limit FDCAN_TTOCF.AWL specifies the number of NTUs between two times the watchdog has to be served. The maximum number of NTUs is 256. The application watchdog is served by reading register FDCAN_TTOST. FDCAN_TTOST.AWE indicates whether the watchdog has been served in time. In case the application failed to serve the application watchdog, interrupt flag FDCAN_TTIR.AW is set. For software development, the application watchdog may be disabled by programming FDCAN_TTOCF.AWL to 0x00, see Section 61.5.49.

**Timing of interface signals**

The timing events that cause a pulse at output FDCAN trigger time mark interrupt pulse fdcan1_tmp for more than one instance and fdcan_tmp if only one instance; FDCAN register time mark interrupt pulse fdcan1_rtp for more than one instance and fdcan_rtp if only one instance are generated in the CAN clock domain.

There is a clock domain crossing delay to be considered before the same event is visible in the APB clock domain (when FDCAN_TTIR.TTMI is set or FDCAN_TTIR.RTMI is set). As an example, the signals can be connected to the timing input(s) of another FDCAN node (fdcan应力fdcan_evt), in order to automatically synchronize two TTCAN networks.

Output FDCAN start of cycle_fdcan1_soc for more than one instance and cycle_fdcan1_soc if only one instance gets active whenever a reference message is completed (either transmitted or received). The output is controlled in the APB clock domain.

**61.4.9 Message scheduling**

FDCAN_TTOCF.TM controls whether the TTCAN operates as potential time master or as a time slave. If it is a potential time master, the three LSBs of the reference message identifier FDCAN_TTRMC.RID define the master priority, 0 giving the highest and 7 giving the lowest. There cannot be two nodes in the network using the same master priority.

FDCAN_TTRMC.RID is used for recognition of reference messages.

FDCAN_TTRMC.RMPS is not relevant for time slaves.

The initial reference trigger offset FDCAN_TTOCF.IRTO is a 7-bit-value that defines (in NTUs) how long a backup time master waits before it starts the transmission of a reference message when a reference message is expected but the bus remains idle. The recommended value for FDCAN_TTOCF.IRTO is the master priority multiplied with a factor depending on the expected clock drift between the potential time masters in the network. The sequential order of the backup time masters, when one of them starts the reference message in case the current time master fails, should correspond to their master priority, even with maximum clock drift.

FDCAN_TTOCF.OM decides whether the node operates in TTCAN level 0, level 1, or level 2. In one network, all potential time masters have to operate on the same level. Time slaves may operate on level 1 in a level 2 network, but not vice versa. The configuration of the TTCAN operation mode via FDCAN_TTOCF.OM is the last step in the setup. With FDCAN_TTOCF.OM = 00 (event-driven CAN communication), the FDCAN operates
according to ISO 11898-1: 2015, without time triggers. With FDCAN_TTOCF.OM = 01 (level 1), the FDCAN operates according to ISO 11898-4, but without the possibility to synchronize the basic cycles to external events, the Next_is_Gap bit in the reference message is ignored. With FDCAN_TTOCF.OM = 10 (level 2), the TTCAN operates according to ISO 11898-4, including the event-synchronized start of a basic cycle. With FDCAN_TTOCF.OM = 11 (level 0), the FDCAN operates as event-driven CAN but maintains a calibrated global time base as in level 2.

FDCAN_TTOCF.EECS enables the external clock synchronization, allowing the application program of the current time master to update the TUR configuration during time-triggered operation, to adapt the clock speed and (in levels 0 and level 2 only) the global clock phase to an external reference.

FDCAN_TTMLM.ENTT in the TT matrix limits register specifies the number of expected Tx_Triggers in the system matrix. This is the sum of Tx_Triggers for exclusive, single arbitrating and merged arbitrating windows, excluding the Tx_Ref_Triggers. Note that this is usually not the number of Tx_Trigger memory elements; the number of basic cycles in the system matrix and the trigger repeat factors have to be taken into account.

An inaccurate configuration of FDCAN_TTMLM.ENTT will result either in a TxCount Underflow (FDCAN_TTIR.TXU = 1 and FDCAN_TTOST.EL = 01, severity 1), or in a Tx count Overflow (FDCAN_TTIR.TXO = 1 and FDCAN_TTOST.EL = 10, severity 2).

Note: In case the first reference message seen by a node does not have Cycle_Count 0, this node may finish its first matrix cycle with its Tx count resulting in a Tx count underflow condition. As long as a node is in state Synchronizing its Tx_Triggers will not lead to transmissions.

FDCAN_TTMLM.CCM specifies the number of the last basic cycle in the system matrix. The counting of basic cycles starts at 0. In a system matrix consisting of eight basic cycles FDCAN_TTMLM.CCM would be 7. FDCAN_TTMLM.CCM is ignored by time slaves, a receiver of a reference message considers the received cycle count as the valid cycle count for the actual basic cycle.

FDCAN_TTMLM.TXEW specifies the length of the Tx enable window in NTUs. The Tx enable window is that period of time at the beginning of a time window where a transmission may be started. If a transmission of a message cannot be started inside the Tx enable window because of for example, a slight overlap from the previous time window message, the transmission cannot be started in that time window at all. FDCAN_TTMLM.TXEW has to be chosen with respect to the network synchronization quality and with respect to the relation between the length of the time windows and the length of the messages.

Trigger memory

The trigger memory is part of the external message RAM to which the TTCAN is connected to (see Section 61.5.26). It stores up to 64 trigger elements. A trigger memory element consists of time mark TM, cycle code CC, trigger type TYPE, filter type FTYPE, message Number MNR, message status count MSC, time mark event internal TMIN, time mark event external TMEX (see Section 61.4.23).

The time mark defines at which cycle time a trigger becomes active. The triggers in the trigger memory have to be sorted by their time marks. The trigger element with the lowest time mark is written to the first trigger memory word. Message number and cycle code are ignored for triggers of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, and End_of_List.

When the cycle time reaches the time mark of the actual trigger, the FSE switches to the next trigger and starts to read the following trigger from the trigger memory. In case of a
transmit trigger, the Tx handler starts to read the message from the message RAM as soon as the FSE switches to its trigger. The RAM access speed defines the minimum time step between a transmit trigger and its preceding trigger, the Tx handler has to be able to prepare the transmission before the transmit trigger time mark is reached. The RAM access speed also limits the number of non-matching (with regard to their cycle code) triggers between two matching triggers, the next matching trigger must be read before its time mark is reached. If the reference message is n NTU long, a trigger with a time mark lower than n will never become active and will be treated as a configuration error.

Starting point of the cycle time is the sample point of the reference message start of frame bit. The next reference message is requested when cycle time reaches the Tx_Ref_Trigger time mark. The FDCAN reacts on the transmission request at the next sample point. A new Sync_Mark is captured at the start of frame bit, but the cycle time is incremented until the reference message is successfully transmitted (or received) and the Sync_Mark is taken as the new Ref_Mark. At that point in time, cycle time is restarted. As a consequence, cycle time can never (with the exception of initialization) be seen at a value lower than n, with n being the length of the reference message measured in NTU. Length of a basic cycle: Tx_Ref_Trigger time mark + 1 NTU + 1 CAN bit time.

The trigger list will be different for all nodes in the CAN FD network. Each node knows only the Tx_Triggers for its own transmit messages, the Rx_Triggers for those receive messages that are processed by this node, and the triggers concerning the reference messages.

**Trigger types**

- **Tx_Ref_Trigger (TYPE = 0000)** and **Tx_Ref_Trigger_Gap (TYPE = 0001)** cause the transmission of a reference message by a time master. A configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when a time slave encounters a Tx_Ref_Trigger(_Gap) in its trigger memory. Tx_Ref_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, Tx_Ref_Trigger is ignored when the FDCAN synchronization state is In_Gap (FDCAN_TTOST.SYS = 10).

- **Tx_Trigger_Single (TYPE = 0010)**, **Tx_Trigger_Continuous (TYPE = 0011)**, **Tx_Trigger_Arbitration (TYPE = 0100)**, and **Tx_Trigger_Merged (TYPE = 0101)** cause the start of a transmission. They define the start of a time window.

  - **Tx_Trigger_Single** starts a single transmission in an exclusive time window when the message buffer transmission request pending bit is set. After successful transmission the transmission request pending bit is reset.

  - **Tx_Trigger_Continuous** starts a transmission in an exclusive time window when the message buffer request pending bit is set. After successful transmission the transmission request pending bit remains set, and the message buffer is transmitted again in the next matching time window.

  - **Tx_Trigger_Arbitration** starts an arbitrating time window, **Tx_Trigger_Merged** a merged arbitrating time window. The last Tx_Trigger of a merged arbitrating time window must be of type Tx_Trigger_Arbitration. A configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when a trigger of type Tx_Trigger_Merged is followed by any other Tx_Trigger than one of type Tx_Trigger_Merged or Tx_Trigger_Arbitration. Several Tx_Triggers may be defined for the same Tx message buffer. Depending on their cycle code, they may be ignored in some basic cycles. The cycle code has to be considered when the expected number of Tx_Triggers (FDCAN_TTMLM.ENTT) is calculated.
Watch_Trigger (TYPE = 0110) and Watch_Trigger_Gap (TYPE = 0111) check for missing reference messages. They are used by both time masters and time slaves. Watch_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, a Watch_Trigger is ignored when the FDCAN synchronization state is In_Gap (FDCAN_TTOST.SYS = 10).

Rx_Trigger (TYPE = 1000) is used to check for the reception of periodic messages in exclusive time windows. Rx_Triggers are not active until state In_Schedule or In_Gap is reached. The time mark of an Rx_Trigger shall be placed after the end of that message transmission, independent of time window boundaries. Depending on their cycle code, Rx_Triggers may be ignored in some basic cycles. At the time mark of the Rx_Trigger, it is checked whether the last received message before this time mark and after start of cycle or previous Rx_Trigger had matched the acceptance filter element referenced by MNR. Accepted messages are stored in one of the two receive FIFOs, according to the acceptance filtering, independent of the Rx_Trigger. Acceptance filter elements referenced by Rx_Triggers should be placed at the beginning of the filter list to ensure that the filtering is finished before the Rx_Trigger time mark is reached.

Time_Base_Trigger (TYPE = 1001) are used to generate internal/external events depending on the configuration of TMIN and TMEX.

End_of_List (TYPE = 1010 … 1111) is an illegal trigger type, a configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when an End_of_List trigger is encountered in the trigger memory before the Watch_Trigger and Watch_Trigger_Gap.

**Restrictions for the node trigger list**

There may not be two triggers that are active at the same cycle time and cycle count, but triggers that are active in different basic cycles (different cycle code) may share the same time mark.

Rx_Triggers and Time_Base_Triggers may not be placed inside the Tx enable windows of Tx_Trigger_Single/Continuous/Arbitration, but they may be placed after Tx_Trigger_Merged.

Triggers that are placed after the Watch_Trigger (or the Watch_Trigger_Gap when FDCAN_TTOST.SYS = 10) will never become active. The watch triggers themselves will not become active when the reference messages are transmitted on time.

All unused trigger memory words (after the Watch_Trigger or after the Watch_Trigger_Gap when FDCAN_TTOST.SYS = 10) must be set to trigger type End_of_List.

A typical trigger list for a potential time master will begin with a number of Tx_Triggers and Rx_Triggers followed by the Tx_Ref_Trigger and the Watch_Trigger. For networks with external event- synchronized time-triggered communication, this is followed by the Tx_Ref_Trigger_Gap and the Watch_Trigger_Gap. The trigger list for a time slave will be the same but without the Tx_Ref_Trigger and the Tx_Ref_Trigger_Gap.

At the beginning of each basic cycle, that is at each reception or transmission of a reference message, the trigger list is processed starting with the first trigger memory element. The FSE looks for the first trigger with a cycle code that matches the current cycle count. The FSE waits until cycle time reaches the trigger time mark and activates the trigger. Afterwards the FSE looks for the next trigger in the list with a cycle code that matches the current cycle count.

Special consideration is needed for the time around Tx_Ref_Trigger and Tx_Ref_Trigger_Gap. In a time master competing for master ship, the effective time mark of
a Tx_Ref_Trigger may be decremented in order to be the first node to start a reference message. In backup time masters the effective time mark of a Tx_Ref_Trigger or Tx_Ref_Trigger_Gap is the sum of its configured time mark and the reference trigger offset FDCAN_TTOCF.IRTO. In case error level 2 is reached (FDCAN_TTOST.EL = 10), the effective time mark is the sum of its time mark and 0x127. No other trigger elements should be placed in this range otherwise it may happen that the time marks appear out of order and are flagged as a configuration error. Trigger elements which are coming after Tx_Ref_Trigger may never become active as long as the reference messages come in time.

There are interdependencies between the following parameters:
- APB clock frequency
- Speed and waiting time for trigger RAM accesses
- Length of the acceptance filter list
- Number of trigger elements
- Complexity of cycle code filtering in the trigger elements
- Offset between time marks of the trigger elements

**Example for trigger handling**

The example shows how the trigger list is derived from a node system matrix. Assumption is that node A is first time master and has knowledge of the section of the system matrix shown in Table 472.

<table>
<thead>
<tr>
<th>Cycle count</th>
<th>Time mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>Tx7</td>
</tr>
<tr>
<td>1</td>
<td>Rx3</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>Tx7</td>
</tr>
<tr>
<td>4</td>
<td>Tx7</td>
</tr>
</tbody>
</table>

The cycle count starts with 0 and runs until 0, 1, 3, 7, 15, 31, 63 (the corresponding number of basic cycles in the system matrix is, respectively, 1, 2, 4, 8, 16, 32, 64). The maximum cycle count is configured by FDCAN_TTMLM.CCM. The cycle code CC is composed of repeat factor (= value of most significant 1) and the number of the first basic cycle in the system matrix (= bit field after most significant 1).

As an example, with a cycle code of 0b0010011 (repeat factor = 16, first basic cycle = 3) and a maximum cycle count of FDCAN_TTMLM.CCM = 0x3F matches occur at cycle counts 3, 19, 35 and 51.

A trigger element consists of time mark TM, cycle code CC, trigger type TYPE, and message Number MNR. For transmission MNR references the Tx buffer number (0 ... 31). For reception MNR references the number of the filter element (0 ... 127) that matched during acceptance filtering. Depending on the configuration of the filter type FTYPE, the 11-bit or 29-bit message ID filter list is referenced.
In addition a trigger element can be configured for generation of time mark event internal TMIN, and time mark event external TMEX. The message status count MSC holds the counter value (0 ... 7) for scheduling errors for periodic messages in exclusive time windows at the point in time when the time mark of the trigger element became active.

Table 473. Trigger list, Node A

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Mark1</td>
<td>0b0000100</td>
<td>Tx_Trigger_Single</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>Mark 1</td>
<td>0b1000000</td>
<td>Rx_Trigger</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Mark 1</td>
<td>0b1000011</td>
<td>Tx_Trigger_Single</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>Mark 3</td>
<td>0b1000001</td>
<td>Tx_Trigger_Merged</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Mark 3</td>
<td>0b1000011</td>
<td>Rx_Trigger</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>Mark 4</td>
<td>0b1000001</td>
<td>Tx_Trigger_Arbitration</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Mark 4</td>
<td>0b1000100</td>
<td>Rx_Trigger</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>Mark 6</td>
<td>N/A</td>
<td>Tx_Ref_Trigger</td>
<td>0 (Ref)</td>
</tr>
<tr>
<td>8</td>
<td>Mark 7</td>
<td>N/A</td>
<td>Watch_Trigger</td>
<td>N/A</td>
</tr>
<tr>
<td>9</td>
<td>N/A</td>
<td>N/A</td>
<td>End_of_List</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Tx_Trigger_Single, Tx_Trigger_Continuous, Tx_Trigger_Merged, Tx_Trigger_Arbitration, Rx_Trigger, and Time_Base_Trigger are only valid for the specified cycle code. For all other trigger types the cycle code is ignored.

The FSE starts the basic cycle with scanning the trigger list starting from 0 until a trigger with time mark higher than cycle time and with its cycle code CC matching the actual cycle count is reached, or a trigger of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, or Watch_Trigger_Gap is encountered.

When the cycle time reached the time mark TM, the action defined by trigger type TYPE and message Number MNR is started. There is an error in the configuration when End_of_List is reached.

At mark 6 the reference message (always TxRef) is transmitted. After transmission of the reference message the FSE returns to the beginning of the trigger list. When the watch trigger at mark 7 is reached, the node was not able to transmit the reference message; error treatment is started.

Detection of configuration errors

A configuration error is signaled via FDCAN_TTOST.EL = 11 (severity 3) when:
- The FSE comes to a trigger in the list with a cycle code that matches the current cycle count but with a time mark that is less than the cycle time.
- The previous active trigger was a Tx_Trigger_Merged and the FSE comes to a trigger in the list with a cycle code that matches the current cycle count but that is neither a
The FSE of a node with FDCAN_TTOCF.TM=0 (time slave) encounters a Tx_Ref_Trigger or a Tx_Ref_Trigger_Gap.

Any time mark placed inside the Tx enable window (defined by FDCAN_TTMLM.TXEW) of a Tx_Trigger with a matching cycle code.

A time mark is placed near the time mark of a Tx_Ref_Trigger and the reference trigger offset FDCAN_TTOST.RTO causes a reversal of their sequential order measured in cycle time.

**TTCAN schedule initialization**

The synchronization to the TTCAN message schedule starts when FDCAN_CCCR.INIT is reset. The TTCAN can operate strictly time-triggered (FDCAN_TTOCF.GEN = 0) or external event-synchronized time-triggered (FDCAN_TTOCF.GEN = 1). All nodes start with cycle time 0 at the beginning of their trigger list with FDCAN_TTOST.SYS = 00 (out of synchronization), no transmission is enabled with the exception of the reference message. Nodes in external event-synchronized time-triggered operation mode will ignore Tx_Ref_Trigger and Watch_Trigger and will use instead Tx_Ref_Trigger_Gap and Watch_Trigger_Gap until the first reference message decides whether a Gap is active.

**Time slaves**

After configuration, a time slave will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message before reaching the Watch_Triggers. When it reaches Init_Watch_Trigger, interrupt flag FDCAN_TTIR.IWT is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to acknowledge or to send error flags). The first received reference message will restart the FSE and the cycle time.

*Note:* Init_Watch_Trigger is not part of the trigger list. It is implemented as an internal counter that counts up to 0xFFFF = maximum cycle time.

When a time slave has received any message but the reference message before reaching the Watch_Triggers, it will assume a fatal error (FDCAN_TTOST.EL = 11, severity 3), set interrupt flag FDCAN_TTIR.WT, switch off its CAN bus output, and enter the bus monitoring mode (FDCAN_CCCR.MON set to 1). In the bus monitoring mode it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

*Note:* To leave the fatal error state, the Host has to set FDCAN_CCCR.INIT = 1. After reset of FDCAN_CCCR.INIT, the node restarts TTCAN communication.

When no error is encountered during synchronization, the first reference message sets FDCAN_TTOST.SYS = 01 (Synchronizing), the second sets the FDCAN synchronization state (depending on its Next_is_Gap bit) to FDCAN_TTOST.SYS = 11 (In_Schedule) or FDCAN_TTOST.SYS = 10 (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

**Potential time masters**

After configuration, a potential time master will start the transmission of a reference message when it reaches its Tx_Ref_Trigger (or its Tx_Ref_Trigger_Gap when in external event-synchronized time-triggered operation). It will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message or transmit the reference message successfully before reaching the Watch_Triggers (assumed reason: all other nodes still in reset or configuration, giving no acknowledge). When it reaches
Init\_Watch\_Trigger, the attempted transmission is aborted, interrupt flag FDCAN\_TTIR.IWT is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). Resetting FDCAN\_TTIR.IWT will re-enable the transmission of reference messages until next time the Init\_Watch\_Trigger condition is met, or another CAN message is received. The FSE will not be restarted by the reception of a reference message.

When a potential time master reaches the Watch\_Triggers after it has received any message but the reference message, it will assume a fatal error (FDCAN\_TTOST.EL = 11, severity 3), set interrupt flag FDCAN\_TTIR.WT, switch off its CAN bus output, and enter the bus monitoring mode (FDCAN\_CCCR.MON set to 1). In bus monitoring mode, it is still able to receive messages, but cannot send any dominant bits and therefore cannot give acknowledge.

When no error is detected during initialization, the first reference message sets FDCAN\_TTOST.SYS = 01 (synchronizing), the second sets the FDCAN synchronization state (depending on its Next\_is\_Gap bit) to FDCAN\_TTOST.SYS = 11 (In\_Schedule) or FDCAN\_TTOST.SYS = 10 (In\_Gap), enabling all Tx\_Triggers and Rx\_Triggers.

A potential time master is current time master (FDCAN\_TTOST.MS = 11) when it was the transmitter of the last reference message, else it is backup time master (FDCAN\_TTOST.MS = 10).

When all potential time masters have finished configuration, the node with the highest time master priority in the network will become the current time master.

### 61.4.10 TTCAN gap control

All functions related to Gap control apply only when the FDCAN is operated in external event synchronized time-triggered mode (FDCAN\_TTOCF.GEN = 1). In this operation mode the FDCAN message schedule may be interrupted by inserting Gaps between the basic cycles of the system matrix. All nodes connected to the CAN network have to be configured for external event- synchronized time-triggered operation.

During a Gap, all transmissions are stopped and the CAN bus remains idle. A Gap is finished when the next reference message starts a new basic cycle. A Gap starts at the end of a basic cycle that itself was started by a reference message with bit Next\_is\_Gap = 1 e.g. Gaps are initiated by the current time master.

The current time master has two options to initiate a Gap. A Gap can be initiated under software control when the application program writes FDCAN\_TTOCN.NIG = 1. The Next\_is\_Gap bit will be transmitted as 1 with the next reference message. A Gap can also be initiated under hardware control when the application program enables the event trigger input pin fdcan\_evt by writing FDCAN\_TTOCN.GCS = 1. When a reference message is started and FDCAN\_TTOCN.GCS is set, a HIGH level at event trigger pin fdcan\_evt will set Next\_is\_Gap = 1.

As soon as that reference message is completed, the FDCAN\_TTOST.WFE bit will announce the Gap to the time master as well as to the time slaves. The current basic cycle will continue until its last time window. The time after the last time window is the Gap time.

For the actual time master and the potential time masters, FDCAN\_TTOST.GSI will be set when the last basic cycle has finished and the Gap time starts. In nodes that are time slaves, bit FDCAN\_TTOST.GSI will remain at 0.

When a potential time master is in synchronization state In\_Gap (FDCAN\_TTOST.SYS = 10), it has four options to intentionally finish a Gap:
1. Under software control by writing FDCAN_TTOCN.FGP = 1.

2. Under hardware control (FDCAN_TTOCN.GCS = 1) an edge from HIGH to LOW at the
   event-trigger input pin fdcan_evt sets FDCAN_TTOCN.FGP and restarts the schedule.

3. The third option is a time-triggered restart. When FDCAN_TTOCN.TMG = 1, the next
   register time mark interrupt (FDCAN_TTIR.RTMI = 1) will set FDCAN_TTOCN.FGP
   and start the reference message.

4. Finally any potential time master will finish a Gap when it reaches its
   Tx_Ref_Trigger_Gap, assuming that the event to synchronize on did not occur in time.

None of these options can cause a basic cycle to be interrupted with a reference message.

Setting of FDCAN_TTOCN.FGP after the Gap time has started will start the transmission of
a reference message immediately and will thereby synchronize the message schedule.
When FDCAN_TTOCN.FGP is set before the Gap time has started (while the basic cycle is
still in progress), the next reference message is started at the end of the basic cycle, at the
Tx_Ref_Trigger – there will be no Gap time in the message schedule.

In strictly time-triggered operation, bit Next_is_Gap = 1 in the reference message will be
ignored, as well as the event-trigger input pin fdcan_evt and the bits FDCAN_TTOCN.NIG,
FDCAN_TTOCN.FGP, and FDCAN_TTOCN.TMG.

61.4.11 Stop watch

The stop watch function enables capturing of FDCAN internal time values (local time, cycle
time, or global time) triggered by an external event.

To enable the stop watch function, the application program first has to define local time,
cycle time, or global time as stop watch source via FDCAN_TTOCN.SWS. When
FDCAN_TTOCN.SWS is different from 00 and TT interrupt register flag FDCAN_TTIR.SWE
is 0, the actual value of the time selected by FDCAN_TTTOCN.SWS will be copied into
FDCAN_TTCPT.SWV on the next rising / falling edge (as configured via
FDCAN_TTOCN.SWP) on stop watch trigger pin fdcan_swt. This will set interrupt flag
FDCAN_TTIR.SWE. After the application program has read FDCAN_TTCPT.SWV, it may
enable the next stop watch event by resetting FDCAN_TTIR.SWE to 0.

61.4.12 Local time, cycle time, global time,
and external clock synchronization

There are two possible levels in time-triggered CAN:
1. Level 1 only provides time-triggered operation using cycle time.
2. Level 2 additionally provides increased synchronization quality, global time and
   external clock synchronization. In both levels, all timing features are based on a local
time base - the local time.

The local time is a 16-bit cyclic counter, it is incremented once each NTU. Internally the NTU
is represented by a 3-bit counter which can be regarded as a fractional part (three binary
digits) of the local time. Generally, the 3-bit NTU counter is incremented eight times each
NTU. If the length of the NTU is shorter than eight CAN clock periods (as may be configured
in level 1, or as a result of clock calibration in level 2), the length of the NTU fraction is
adapted, and the NTU counter is incremented only four times each NTU.

Figure 711 describes the synchronization of the cycle time and global time, performed in the
same manner by all FDCAN nodes, including the time master. Any message received or
transmitted invokes a capture of the local time taken at the message is frame
synchronization event. This frame synchronization event occurs at the sample point of each start of frame (SoF) bit and causes the local time to be stored as Sync_Mark. Sync_Marks and Ref_Marks are captured including the 3-bit fractional part.

Whenever a valid reference message is transmitted or received, the internal Ref_Mark is updated from the Sync_Mark. The difference between Ref_Mark and Sync_Mark is the cycle sync mark (cycle sync mark = Sync_Mark - Ref_Mark) stored in register FDCAN_TTCSM. The most significant 16 bits of the difference between Ref_Mark and the actual value of the local time is the cycle time (cycle time = local time - Ref_Mark).

**Figure 711. Cycle time and global time synchronization**

The cycle time that can be read from FDCAN_TTCTC.CT is the difference of the node local time and Ref_Mark, both synchronized into the APB clock domain and truncated to 16 bit.

The global time exists for TTCAN level 0 and level 2 only, in level 1 it is invalid. The node view of the global time is the local image of the global time in (local) NTUs. After configuration, a potential time master will use its own local time as global time. The time master establishes its own local time as global time by transmitting its own Ref_Marks as Master_Ref_Marks in the reference message (bytes 3 and 4). The global time that can be read from FDCAN_TTLGT.GT is the sum of the node local time and its local offset, both synchronized into the APB clock domain and truncated to 16 bit. The fractional part is used for clock synchronization only.

A node that receives a reference message calculates its local offset to the global time by comparing its local Ref_Mark with the received Master_Ref_Mark (see Figure 712). The node view of the global time is local time plus local offset. In a potential time master that has never received another time master reference message, Local_Offset will be 0. When a node becomes the current time master after first having received other reference messages,
Local_Offset will be frozen at its last value. In the time receiving nodes, Local_Offset may be subject to small adjustments, due to clock drift, when another node becomes time master, or when there is a global time discontinuity, signaled by Disc_Bit in the reference message. With the exception of global time discontinuity, the global time provided to the application program by register FDCAN_TTLGT is smoothed by a low-pass filtering to have a continuous monotonic value.

**Figure 712. TTCAN level 0 and level 2 drift compensation**

*Figure 712* describes how in TTCAN levels 0 and 2 each time receiving node compensates the drift between its own local clock and the time master clock by comparing the length of a basic cycle in local time and in global time. If there is a difference between the two values and the Disc_Bit in the reference message is not set, a new value for FDCAN_TURNA.NAV is calculated. If the synchronization deviation \( SD = | NC - FDCAN\_TURNNA\_NAV | \leq SDL \) (synchronization deviation limit), the new value for FDCAN_TURNA.NAV takes effect. Else the automatic drift compensation is suspended.

In TTCAN level 0 and level 2, FDCAN_TTOST.QCS indicates whether the automatic drift compensation is active or suspended. In TTCAN level 1, FDCAN_TOST.QCS is always 1.

The current time master may synchronize its local clock speed and the global time phase to an external clock source. This is enabled by bit FDCAN_TTOCF.EECS.

The stop watch function (see Section 61.4.11) may be used to measure the difference in clock speed between the local clock and the external clock. The local clock speed is adjusted by first writing the newly calculated numerator configuration low to
FDCAN_TURCF.NCL (FDCAN_TURCF.DC cannot be updated during operation). The new value takes effect by writing FDCAN_TTOCN.ECS to 1.

The global time phase is adjusted by first writing the phase offset into the TT global time Preset register FDCAN_TTGTP. The new value takes effect by writing FDCAN_TTOCN.SGT to 1. The first reference message transmitted after the global time phase adjustment will have the Disc_Bit set to 1.

FDCAN_TTOST.QGTP shows whether the node global time is in phase with the time master global time. FDCAN_TTOST.QGTP is permanently 0 in TTCAN level 1 and when the synchronization deviation limit is exceeded in TTCAN level 0, 2 (FDCAN_TTOST.QCS = 0). It is temporarily 0 while the global time is low-pass filtered to supply the application with a continuous monotonic value. There is no low-pass filtering when the last reference message contained a Disc_Bit = 1 or when FDCAN_TTOST.QCS = 0.

61.4.13 TTCAN error level

The ISO 11898-4 specifies four levels of error severity:

1. S0 - No error
2. S1 - Warning - Only notification of application, reaction application-specific.
3. S2 error - Notification of application. All transmissions in exclusive or arbitrating time windows are disabled (i.e. no data or remote frames may be started). Potential time masters still transmit reference messages with the reference trigger offset FDCAN_TTOST.RTO set to the maximum value of 127.
4. S3 - Severe error - Notification of application. All CAN bus operations are stopped, i.e. transmission of dominant bits is not allowed, and FDCAN_CCCR.MON is set. The S3 error condition remains active until the application updates the configuration (set FDCAN_CCCR.CCE).

If several errors are detected at the same time, the highest severity prevails. When an error is detected, the application is notified by FDCAN_TTIR.ELC. The error level is monitored by FDCAN_TTOST.EL.

The TTCAN signals the following error conditions as required by ISO 11898-4:

- **Config_error (S3)**
  - Sets error level FDCAN_TTOST.EL to 11 when a merged arbitrating time window is not properly closed or when there is a Tx_Trigger with a time mark beyond the Tx_Ref_Trigger.

- **Watch_Trigger_Reached (S3)**
  - Sets error level FDCAN_TTOST.EL to 11 when a watch trigger was reached because the reference message is missing.

- **Application_Watchdog (S3)**
  - Sets error level FDCAN_TTOST.EL to 11 when the application failed to serve the application watchdog. The application watchdog is configured via FDCAN_TTOCF.AWL. It is served by reading register FDCAN_TTOST. When the watchdog is not served in time, bit FDCAN_TTOST.AWE and interrupt flag
Controller area network with flexible data rate (FDCAN)  

FDCAN_TTIR.AW are set, all FDCAN communication is stopped, and the FDCAN is set into bus monitoring mode (FDCAN_CCCR.MON set to 1).

- **CAN_Bus_Off (S3)**
  - Entering CAN_Bus_Off state sets error level FDCAN_TTOST.EL to 11. CAN_Bus_Off state is signaled by FDCAN_PSR.BO = 1 and FDCAN_CCCR.INIT = 1.

- **Scheduling_Error_2 (S2)**
  - Sets error level FDCAN_TTOST.EL to 10 if the MSC of one Tx_Trigger has reached 7. In addition interrupt flag FDCAN_TTIR.SE2 is set. The error level FDCAN_TTOST.EL is reset to 00 at the beginning of a matrix cycle when no Tx_Trigger has an MSC of 7 in the preceding matrix cycle.

- **Tx_Overflow (S2)**
  - Sets error level FDCAN_TTOST.EL to 10 when the Tx count is equal or higher than the expected number of Tx_Triggers FDCAN_TTMLM.ENTT and a Tx_Trigger event occurs. In addition interrupt flag FDCAN_TTIR.TXO is set. The error level FDCAN_TTOST.EL is reset to 00 when the Tx count is no more than FDCAN_TTMLM.ENTT at the start of a new matrix cycle.

- **Scheduling_Error_1 (S1)**
  - Sets error level FDCAN_TTOST.EL to 01 if within one matrix cycle the difference between the maximum MSC and the minimum MSC for all trigger memory elements (of exclusive time windows) is larger than two, or if one of the MSCs of an exclusive Rx_Trigger has reached seven. In addition interrupt flag FDCAN_TTIR.SE1 is set. If within one matrix cycle none of these conditions is valid, the error level FDCAN_TTOST.EL is reset to 00.

- **Tx_Underflow (S1)**
  - Sets error level FDCAN_TTOST.EL to 01 when the Tx count is less than the expected number of Tx_Triggers FDCAN_TTMLM.ENTT at the start of a new matrix cycle. In addition interrupt flag FDCAN_TTIR.TXU is set. The error level FDCAN_TTOST.EL is reset to 00 when the Tx count is at least FDCAN_TTMLM.ENTT at the start of a new matrix cycle.

### 61.4.14 TTCAN message handling

**Reference message**

For potential time masters the identifier of the reference message is configured via FDCAN_TTRMC.RID. No dedicated Tx buffer is required for transmission of the reference message. When a reference message is transmitted, the first data byte for TTCAN level 1 (that is, the first four data bytes for TTCAN level 0 and the first four data bytes for TTCAN level 2) will be provided by the FSE.

In case the reference message Payload select FDCAN_TTRMC.RMPS is set, the rest of the reference message payload (level 1: bytes 2-8, level 0, 2: bytes 5-6) is taken from Tx buffer 0. In this case the data length DLC code from message buffer 0 is used.

**Table 474. Number of data bytes transmitted with a reference message**

<table>
<thead>
<tr>
<th>FDCAN_TTRMC.RMPS</th>
<th>FDCAN_TXBRP.TRPO</th>
<th>Level 0</th>
<th>Level 1</th>
<th>Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
To send additional payload with the reference message in level 1 a DLC > 1 has to be configured, for level 0 and level 2 a DLC > 4 is required. In addition the transmission request pending bit FDCAN_TXBRP.TRP0 of message buffer 0 must be set (see Table 474). In case bit FDCAN_TXBRP.TRP0 is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.

For acceptance filtering of reference messages the reference identifier FDCAN_TTRMC.RID is used.

**Message reception**

Message reception is done via the two Rx FIFOs in the same way as for event-driven CAN communication (see Rx handler). The message status count MSC is part of the corresponding trigger memory element and has to be initialized to 0 during configuration. It is updated while the TTCAN is in synchronization states In_Gap or In_Schedule. The update happens at the message Rx_Trigger. At this point in time it is checked at which acceptance filter element the latest message received in this basic cycle had matched. The matching filter number is stored as the acceptance filter result. If this is the same the filter number as defined in this trigger memory element, the MSC is decremented by one. If the acceptance filter result is not the same filter number as defined for this filter element, or if the acceptance filter result is cleared, the MSC is incremented by one. At each Rx_Trigger and at each start of cycle, the last acceptance filter result is cleared.

The time mark of an Rx_Trigger should be set to a value where it is ensured that reception and acceptance filtering for the targeted message has completed. This has to take into consideration the RAM access time and the order of the filter list. It is recommended, that filters which are used for Rx_Triggers are placed at the beginning of the filter list. It is not recommended to use an Rx_Trigger for the reference message.

**Message transmission**

For time-triggered message transmission the TTCAN supplies 32 dedicated Tx buffers (see Transmit pause). A Tx FIFO or Tx queue is not available when the FDCAN is configured for time-triggered operation (FDCAN_TTOCF.OM = 01 or 10).

Each Tx_Trigger in the trigger memory points to a particular Tx buffer containing a specific message. There may be more than one Tx_Trigger for a given Tx buffer if that Tx buffer contains a message that is to be transmitted more than once in a basic cycle or matrix cycle.

The application program has to update the data regularly and on time, synchronized to the cycle time. The Host CPU is responsible that no partially updated messages are transmitted. To assure this the Host has to proceed in the following way:

<table>
<thead>
<tr>
<th>FDCAN_TTRMC.RMPS</th>
<th>FDCAN_TXBRP.TRP0</th>
<th>Level 0</th>
<th>Level 1</th>
<th>Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4 + MBO</td>
<td>1 + MBO</td>
<td>4 + MBO</td>
</tr>
</tbody>
</table>

To send additional payload with the reference message in level 1 a DLC > 1 has to be configured, for level 0 and level 2 a DLC > 4 is required. In addition the transmission request pending bit FDCAN_TXBRP.TRP0 of message buffer 0 must be set (see Table 474). In case bit FDCAN_TXBRP.TRP0 is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.
**Tx_Trigger_Single / Tx_Trigger_Merged / Tx_Trigger_Arbitration**
- Check whether the previous transmission has completed by reading FDCAN_TXBTO
- Update the Tx buffer configuration and/or payload
- Issue an add request to set the Tx buffer request pending bit

**Tx_Trigger_Continuous**
- Issue a cancellation request to reset the Tx buffer request pending bit
- Check whether the cancellation has finished by reading FDCAN_TXBCF
- Update Tx buffer configuration and/or payload
- Issue an add request to set the Tx buffer request pending bit

The message MSC stored with the corresponding Tx_Trigger provides information on the success of the transmission.

The MSC is incremented by one when the transmission could not be started because the CAN bus was not idle within the corresponding transmit enable window or when the message was started and could not be completed successfully. The MSC is decremented by one when the message was transmitted successfully or when the message could have been started within its transmit enable window but was not started because transmission was disabled (TTCAN in error level S2 or Host has disabled this particular message).

The Tx buffers may be managed dynamically, i.e. several messages with different identifiers may share the same Tx buffer element. In this case the Host has to assure that no transmission request is pending for the Tx buffer element to be reconfigured by checking FDCAN_TXBRP.

If a Tx buffer with pending transmission request should be updated, the Host first has to issue a cancellation request and check whether the cancellation has completed by reading FDCAN_TXBCF before it starts updating.

The Tx handler will transfer a message from the message RAM to its intermediate output buffer at the trigger element which becomes active immediately before the Tx_Trigger element which defines the beginning of the transmit window. During and after the transfer time the transmit message may not be updated and its FDCAN_TXBRP bit may not be changed. To control this transfer time, an additional trigger element may be placed before the Tx_Trigger. This may be example of a Time_Base_Trigger which need not cause any other action. The difference in time marks between the Tx_Trigger and the preceding trigger has to be large enough to guarantee that the Tx handler can read four words from the message RAM even at high RAM access load from other modules.

**Transmission in exclusive time windows**

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Single or Tx_Trigger_Continuous. There is no arbitration on the bus with messages from other nodes. The MSC is updated according the result of the transmission attempt. After successful transmission started by a Tx_Trigger_Single the respective Tx buffer request pending bit is reset. After successful transmission started by a Tx_Trigger_Continuous the respective Tx buffer request pending remains set. When the transmission was not successful due to disturbances, it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.
Transmission in arbitrating time windows
A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Arbitration. Several nodes may start to transmit at the same time. In this case the message has to arbitrate with the messages from other nodes. The MSC is not updated. When the transmission was not successful (lost arbitration or disturbance), it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

Transmission in merged arbitrating time windows
The purpose of a merged arbitrating time window is, to enable multiple nodes to send a limited number of frames which are transmitted in immediate sequence, the order given by CAN arbitration. It is not intended for burst transmission by a node. Since the node does not have exclusive access within this time window, it may happen that not all requested transmissions are successful.

Messages which have lost arbitration or were disturbed by an error, may be re-transmitted inside the same merged arbitrating time window. The re-transmission will not be started if the corresponding transmission request pending flag was reset by a successful Tx cancellation.

In single transmit windows, the Tx handler transmits the message indicated by the message number of the trigger element. In merged arbitrating time windows, it can handle up to three message numbers from the trigger list. Their transmissions will be attempted in the sequence defined by the trigger list. If the time mark of a fourth message is reached before the first is transmitted (or canceled by the Host), the four the request will be ignored.

The transmission inside a merged arbitrating time window is not time-triggered. The transmission of a message may start before its time mark, or after the time mark if the bus was not idle.

The messages transmitted by a specific node inside a merged arbitrating time window will be started in the order of their Tx_Triggers, so a message with low CAN priority may prevent the successful transmission of a following message with higher priority, if their is compelling bus traffic. This has to be considered for the configuration of the trigger list. Time_Base_Triggers may be placed between consecutive Tx_Triggers to define the time until the data of the corresponding Tx buffer needs to be updated.

61.4.15 TTCAN interrupt and error handling
The TT interrupt register FDCAN_TTIR consists of our segments. Each interrupt can be enabled separately by the corresponding bit in the TT interrupt enable register FDCAN_TTIE. The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position.

The first segment consists of flags CER, AW, WT, and IWT. Each flag indicates a fatal error condition where the CAN communication is stopped. With the exception of IWT, these error conditions require a re-configuration of the FDCAN module before the communication can be restarted.

The second segment consists of flags ELC, SE1, SE2, TXO, TXU, and GTE. Each flag indicates an error condition where the CAN communication is disturbed. If they are caused by a transient failure, e.g. by disturbances on the CAN bus, they will be handled by the FDCAN protocol failure handling and do not require intervention by the application program.

The third segment consists of flags GTD, GTW, SWE, TTMI, and RTMI. The first two flags are controlled by global time events (level 0, 2 only) that require a reaction by the application
Controller area network with flexible data rate (FDCAN) program. With a stop watch event triggered by a rising edge on pin fdcan_swt internal time values are captured. The trigger time mark interrupt notifies the application that a specific Time_Base_Trigger is reached. The register time mark interrupt signals that the time referenced by FDCAN_TTOCN.TMC (cycle, local, or global) equals time mark FDCAN_TTTMK.TM. It can also be used to finish a Gap.

The fourth segment consists of flags SOG, CSM, SMC, and SBC. These flags provide a means to synchronize the application program to the communication schedule.

61.4.16 Level 0

TTCAN level 0 is not part of ISO11898-4. This operation mode makes the hardware, that in TTCAN level 2 maintains the calibrated global time base, also available for event-driven CAN according to ISO11898-1.

Level 0 operation is configured via FDCAN_TTOCF.OM = 11. In this mode the FDCAN operates in event driven CAN communication, there is no fixed schedule, the configuration of FDCAN_TTOCF:GEN is ignored. External event-synchronized operation is not available in level 0. A synchronized time base is maintained by transmission of reference messages.

In level 0 the trigger memory is not active and therefore needs not to be configured. The time mark interrupt flag (FDCAN_TTIR.TTMI) is set when the cycle time has reached FDCAN_TTOCF.IRTO `0x200, it reminds the Host to set a transmission request for message buffer 0. The Watch_Trigger interrupt flag (FDCAN_TTIR.WT) is set when the cycle time has reached 0xFF00. These values were chosen to have enough margin for a stable clock calibration. There are no further TT-error-checks.

Register time mark interrupts (FDCAN_TTIR.RTMI) are also possible.

The reference message is configured as for level 2 operation. Received reference messages are recognized by the identifier configured in register FDCAN_TTRMC. For the transmission of reference messages only message buffer 0 may be used. The node transmits reference messages any time the Host sets a transmission request for message buffer 0, there is no reference trigger offset.

Level 0 operation is configured via:

- FDCAN_TTRMC
- FDCAN_TTOCF except EVTP, AWL, GEN
- FDCAN_TTMLM except ENTT, TXEW
- FDCAN_TURCF

Level 0 operation is controlled via:

- FDCAN_TTOCN except NiG, TMG, FGP, GCS, TTMIE
- FDCAN_TTGTP
- FDCAN_TTTMK
- FDCAN_TTIR excluding bits CER, AW, IWT SE2, SE1, TXO, TXU, SOG (no function)
- FDCAN_TTIR the following bits have changed function
  - TTMI not defined by trigger memory - activated at cycle time FDCAN_TTOCF.IRTO 0x200
  - WT not defined by trigger memory - activated at cycle time 0xFF00

Level 0 operation is signaled via:

- TTOST excluding bits AWE, WFE, GSI, GFI, RTO (no function)
Synchronizing

Figure 713 describes the states and the state transitions in TTCAN level 0 operation. Level 0 has no InGap state.

Figure 713. Level 0 schedule synchronization state machine

Handling of error levels

During level 0 operation only the following error conditions may occur:

- Watch_Trigger_Reached (S3), reached cycle time 0xFF00
- CAN_Bus_Off (S3)

Since no S1 and S2 errors are possible, the error level can only switch between S0 (No error) and S3 (Severe error). In TTCAN level 0 an S3 error is handled differently. When error level S3 is reached, both FDCAN_TTOST.SYS and FDCAN_TTOST.MS are reset, and interrupt flags FDCAN_TTIR.GTE and FDCAN_TTIR.GTD are set.

When error level S3 (FDCAN_TTOST.EL = 11) is entered, bus monitoring mode is (contrary to TTCAN level 1 and level 2) not entered. S3 error level is left automatically after transmission (time master) or reception (time slave) of the next reference message.
Master slave relation

*Figure 714* describes the master slave relation in TTCAN level 0. In case of an S3 error the FDCAN returns to state Master_Off.

**Figure 714. Level 0 master to slave relation**

61.4.17 Synchronization to external time schedule

This feature can be used to synchronize the phase of the FDCAN schedule to an external schedule (e.g. that of a second TTCAN network or FlexRay network). It is applicable only when the FDCAN is current time master (FDCAN_TTOST.MS = 11).

External synchronization is controlled by event trigger input pin fdcan_evt. If bit FDCAN_TTOCN.ESCN is set, a rising edge at event trigger pin fdcan_evt the FDCAN compares its actual cycle time with the target phase value configured by FDCAN_TTGTP.CTP.

Before setting FDCAN_TTOCN.ESCN the Host has to adapt the phases of the two time schedules e.g. by using the FDCAN gap control (see *Section 61.4.10*). When the Host sets FDCAN_TTOCN.ESCN, FDCAN_TTOSTSPL is set.

If the difference between the cycle time and the target phase value FDCAN_TTGTP.CTP at the rising edge at event trigger pin fdcan_evt is greater than 9 NTU, the phase lock bit FDCAN_TTOST.SPL is reset, and interrupt flag FDCAN_TTIR.CSM is set.
FDCAN_TTOST.SPL is also reset (and FDCAN_TTIR.CSM is set), when another node becomes time master.

If both FDCAN_TTOST.SPL and FDCAN_TTOCN.ESCN are set, and if the difference between the cycle time and the target phase value FDCAN_TTGTP.CTP at the rising edge at event trigger pin fdcan_evt is lower or equal to nine NTU, the phase lock bit FDCAN_TTOST.SPL remains set, and the measured difference is used as reference trigger offset value to adjust the phase at the next transmitted reference message.

Note: The rising edge detection at event trigger pin fdcan_evt is enabled with the start of each basic cycle. The first rising edge triggers the compare of the actual cycle time with FDCAN_TTGTP.CTP. All further edges until the beginning of the next basic cycle are ignored.

61.4.18 FDCAN Rx buffer and FIFO element

Up to 64 Rx buffers and two Rx FIFOs can be configured in the message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx buffer / FIFO element is shown in Table 475, the description is provided in Table 476.

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>24</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>ESI</td>
<td>XTD</td>
<td>RTR</td>
<td>ID[28:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>ANMF</td>
<td>FIDX[6:0]</td>
<td>Res.</td>
<td>FDF</td>
<td>BRS</td>
<td>DLC[3:0]</td>
<td>RXTS[15:0]</td>
</tr>
<tr>
<td>R2</td>
<td>DB3[7:0]</td>
<td>DB2[7:0]</td>
<td>DB1[7:0]</td>
<td>DB0[7:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>DB7[7:0]</td>
<td>DB6[7:0]</td>
<td>DB5[7:0]</td>
<td>DB4[7:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rn</td>
<td>DBm[7:0]</td>
<td>DBm-1[7:0]</td>
<td>DBm-2[7:0]</td>
<td>DBm-3[7:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register FDCAN_RXESC.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| R0 bit 31 ESI | Error state indicator  
0: Transmitting node is error active  
1: Transmitting node is error passive |
| R0 bit 30 XTD | Extended identifier  
Signals to the Host whether the received frame has a standard or extended identifier.  
0: 11-bit standard identifier  
1: 29-bit extended identifier |
| R0 bit 29 RTR | Remote transmission request  
Signals to the Host whether the received frame is a data frame or a remote frame.  
0: Received frame is a data frame  
1: Received frame is a remote frame |
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0 bits 28:0</td>
<td>Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].</td>
</tr>
<tr>
<td>R1 bit 31 ANMF</td>
<td>Accepted non-matching frame Acceptance of non-matching frames may be enabled via FDCAN_GFC.ANFS and FDCAN_GFC.ANFE. 0: Received frame matching filter index FIDX 1: Received frame did not match any Rx filter element</td>
</tr>
<tr>
<td>R1 bits 30:24 FIDX[6:0]</td>
<td>Filter index 0-127 = index of matching Rx acceptance filter element (invalid if ANMF = 1). Range is 0 to FDCAN_SIDFC.LSS. - 1 or FDCAN_XIDFC.LSE. - 1.</td>
</tr>
<tr>
<td>R1 bit 21 FDF</td>
<td>FD Format 0: Standard frame format 1: FDCAN frame format (new DLC-coding and CRC)</td>
</tr>
<tr>
<td>R1 bit 20 BRS</td>
<td>Bitrate Switch 0: Frame received without bitrate switching 1: Frame received with bitrate switching</td>
</tr>
<tr>
<td>R1 bits 19:16 DLC[3:0]</td>
<td>Data length code 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes 9-15: Classic CAN: received frame has 8 data bytes 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes</td>
</tr>
<tr>
<td>R1 bits 15:0 RXTS[15:0]</td>
<td>Rx timestamp Timestamp counter value captured on start of frame reception. Resolution depending on configuration of the timestamp counter prescaler FDCAN_TSCC.TCP.</td>
</tr>
<tr>
<td>R2 bits 31:24 DB3[7:0]</td>
<td>Data Byte 3</td>
</tr>
<tr>
<td>R2 bits 23:16 DB2[7:0]</td>
<td>Data Byte 2</td>
</tr>
<tr>
<td>R2 bits 15:8 DB1[7:0]</td>
<td>Data Byte 1</td>
</tr>
<tr>
<td>R2 bits 7:0 DB0[7:0]</td>
<td>Data Byte 0</td>
</tr>
<tr>
<td>R3 bits 31:24 DB7[7:0]</td>
<td>Data Byte 7</td>
</tr>
<tr>
<td>R3 bits 23:16 DB6[7:0]</td>
<td>Data Byte 6</td>
</tr>
<tr>
<td>R3 bits 15:8 DB5[7:0]</td>
<td>Data Byte 5</td>
</tr>
<tr>
<td>R3 bits 7:0 DB4[7:0]</td>
<td>Data Byte 4</td>
</tr>
</tbody>
</table>

...
61.4.19 FDCAN Tx buffer element

The Tx buffers section can be configured to hold dedicated Tx buffers as well as a Tx FIFO / Tx queue. In case that the Tx buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx queue, the dedicated Tx buffers start at the beginning of the Tx buffers section followed by the buffers assigned to the Tx FIFO or Tx queue. The Tx handler distinguishes between dedicated Tx buffers and Tx FIFO / Tx queue by evaluating the Tx buffer configuration FDCAN_TXBC.TFQS and FDCAN_TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register FDCAN_TXESC.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ln bits 31:24</td>
<td></td>
</tr>
<tr>
<td>DBn[7:0]</td>
<td>Data Byte m</td>
</tr>
<tr>
<td>Ln bits 23:16</td>
<td></td>
</tr>
<tr>
<td>DBm-1[7:0]</td>
<td>Data Byte m-1</td>
</tr>
<tr>
<td>Ln bits 15:8</td>
<td></td>
</tr>
<tr>
<td>DBm-2[7:0]</td>
<td>Data Byte m-2</td>
</tr>
<tr>
<td>Ln bits 7:0</td>
<td></td>
</tr>
<tr>
<td>DBm-3[7:0]</td>
<td>Data Byte m-3</td>
</tr>
</tbody>
</table>

Table 477. Tx buffer and FIFO element

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>ESI</td>
<td>XTD</td>
<td>RTR</td>
<td>ID[28:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>MM[7:0]</td>
<td>EFC</td>
<td>Res.</td>
<td>FDF</td>
<td>BPS</td>
<td>DLC[3:0]</td>
<td>Reserved</td>
</tr>
<tr>
<td>T2</td>
<td>DB3[7:0]</td>
<td>DB2[7:0]</td>
<td>DB1[7:0]</td>
<td>DB0[7:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>DB7[7:0]</td>
<td>DB6[7:0]</td>
<td>DB5[7:0]</td>
<td>DB4[7:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tn</td>
<td>DBm[7:0]</td>
<td>DBm-1[7:0]</td>
<td>DBm-2[7:0]</td>
<td>DBm-3[7:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 478. Tx buffer element description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0 bit 31 ESI(1)</td>
<td>Error state indicator 0: ESI bit in CAN FD format depends only on error passive flag 1: ESI bit in CAN FD format transmitted recessive</td>
</tr>
<tr>
<td>T0 bit 30 XTD</td>
<td>Extended identifier 0: 11-bit standard identifier 1: 29-bit extended identifier</td>
</tr>
<tr>
<td>T0 bit 29 RTR(2)</td>
<td>Remote transmission request 0: Transmit data frame 1: Transmit remote frame</td>
</tr>
</tbody>
</table>
### Table 478. Tx buffer element description (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| T0 bits 28:0 ID[28:0] | Identifier  
Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18]. |
| T1 bits 31:24 MM[7:0] | Message marker  
Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status. |
| T1 bit 23 EFC | Event FIFO control  
0: Don’t store Tx events  
1: Store Tx events |
| T1 bit 21 FDF | FD Format  
0: Frame transmitted in classic CAN format  
1: Frame transmitted in CAN FD format |
| T1 bit 20 BRS(3) | Bitrate switching  
0: CAN FD frames transmitted without bitrate switching  
1: CAN FD frames transmitted with bitrate switching |
| T1 bits 19:16 DLC[3:0] | Data length code  
0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes  
9-15: Classic CAN: received frame has 8 data bytes  
9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes |
| T2 bits 31:24 DB3[7:0] | Data Byte 3 |
| T2 bits 23:16 DB2[7:0] | Data Byte 2 |
| T2 bits 15:8 DB1[7:0] | Data Byte 1 |
| T2 bits 7:0 DB0[7:0] | Data Byte 0 |
| T3 bits 31:24 DB7[7:0] | Data Byte 7 |
| T3 bits 23:16 DB6[7:0] | Data Byte 6 |
| T3 bits 15:8 DB5[7:0] | Data Byte 5 |
| T3 bits 7:0 DB4[7:0] | Data Byte 4 |
| ... | ... |
| Tn bits 31:24 DBm[7:0] | Data Byte m |
| Tn bits 23:16 DBm-1[7:0] | Data Byte m-1 |
### 61.4.20 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx event FIFO can be obtained from register FDCAN_TXEFS.

#### Table 479. Tx Event FIFO element

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>ESI</td>
<td>XTD</td>
<td>RTR</td>
<td>ID[28:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Table 480. Tx Event FIFO element description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| E0 bit 31 ESI | Error state indicator  
  – 0: Transmitting node is error active  
  – 1: Transmitting node is error passive |
| E0 bit 30 XTD | Extended Identifier  
  – 0: 11-bit standard identifier  
  – 1: 29-bit extended identifier |
| E0 bit 29 RTR | Remote transmission request  
  – 0: Transmit data frame  
  – 1: Transmit remote frame |
| E0 bits 28:0 ID[28:0] | Identifier  
  Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18]. |
| E1 bits 31:24 MM[7:0] | Message marker  
  Copied from Tx buffer into Tx event FIFO element for identification of Tx message status. |
61.4.21 FDCAN standard message ID filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is the filter list standard start address FDCAN_SIDFC.FLSSA plus the index of the filter element (0 … 127).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| E1 bits 23:22 EFC | Event type  
– 00: Reserved  
– 01: Tx event  
– 10: Transmission in spite of cancellation (always set for transmissions in DAR mode)  
– 11: Reserved |
| E1 bit 21 EDL | Extended data length  
– 0: Standard frame format  
– 1: FDCAN frame format (new DLC-coding and CRC) |
| E1 bit 20 BRS | Bitrate switching  
– 0: Frame transmitted without bitrate switching  
– 1: Frame transmitted with bitrate switching |
| T1 bits 19:16 DLC[3:0] | Data length code  
– 0 - 8: Frame with 0-8 data bytes transmitted  
– 9 - 15: Frame with eight data bytes transmitted |
| E1 bits 15:0 TXTS[15:0] | Tx timestamp  
Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the timestamp counter prescaler FDCAN_TSCC.TCP. |

Table 481. Standard message ID filter element

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
</table>
### Table 482. Standard message ID filter element field description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bit 31:30</strong></td>
<td><strong>SF[1:0]</strong>(1)</td>
</tr>
<tr>
<td></td>
<td>Standard filter type</td>
</tr>
<tr>
<td></td>
<td>- 00: Range filter from SFID1 to SFID2</td>
</tr>
<tr>
<td></td>
<td>- 01: Dual ID filter for SFID1 or SFID2</td>
</tr>
<tr>
<td></td>
<td>- 10: Classic filter: SFID1 = filter, SFID2 = mask</td>
</tr>
<tr>
<td></td>
<td>- 11: Filter element disabled</td>
</tr>
<tr>
<td><strong>Bit 29:27</strong></td>
<td><strong>SFEC[2:0]</strong></td>
</tr>
<tr>
<td></td>
<td>Standard filter element configuration</td>
</tr>
<tr>
<td></td>
<td>All enabled filter elements are used for acceptance filtering of standard</td>
</tr>
<tr>
<td></td>
<td>frames. Acceptance filtering stops at the first matching enabled filter</td>
</tr>
<tr>
<td></td>
<td>element or when the end of the filter list is reached. If SFEC = “100”,</td>
</tr>
<tr>
<td></td>
<td>“101”, or “110” a match sets interrupt flag FDCAN_IR.HPM and, if enabled,</td>
</tr>
<tr>
<td></td>
<td>an interrupt is generated. In this case register FDCAN_HPMS is updated with</td>
</tr>
<tr>
<td></td>
<td>the status of the priority match.</td>
</tr>
<tr>
<td></td>
<td>- 000: Disable filter element</td>
</tr>
<tr>
<td></td>
<td>- 001: Store in Rx FIFO 0 if filter matches</td>
</tr>
<tr>
<td></td>
<td>- 010: Store in Rx FIFO 1 if filter matches</td>
</tr>
<tr>
<td></td>
<td>- 011: Reject ID if filter matches</td>
</tr>
<tr>
<td></td>
<td>- 100: Set priority if filter matches</td>
</tr>
<tr>
<td></td>
<td>- 101: Set priority and store in FIFO 0 if filter matches</td>
</tr>
<tr>
<td></td>
<td>- 110: Set priority and store in FIFO 1 if filter matches</td>
</tr>
<tr>
<td></td>
<td>- 111: Store into Rx buffer or as debug message, configuration of FDCAN_SFT[1:0] ignored</td>
</tr>
<tr>
<td><strong>Bits 26:16</strong></td>
<td><strong>SFID1[10:0]</strong></td>
</tr>
<tr>
<td></td>
<td>Standard filter ID 1</td>
</tr>
<tr>
<td></td>
<td>First ID of standard ID filter element</td>
</tr>
<tr>
<td></td>
<td>When filtering for Rx buffers or for debug messages this field defines the ID</td>
</tr>
<tr>
<td></td>
<td>of a standard message to be stored. The received identifiers must match</td>
</tr>
<tr>
<td></td>
<td>exactly, no masking mechanism is used.</td>
</tr>
<tr>
<td><strong>Bits 15:0</strong></td>
<td><strong>SFID2[15:10]</strong></td>
</tr>
<tr>
<td></td>
<td>Standard filter ID 2</td>
</tr>
<tr>
<td></td>
<td>This bit field has a different meaning depending on the configuration of</td>
</tr>
<tr>
<td></td>
<td>SFEC:</td>
</tr>
<tr>
<td></td>
<td>- SFEC = 001 ... 110 Second ID of standard ID filter element</td>
</tr>
<tr>
<td></td>
<td>- SFEC = 111 Filter for Rx buffers or for debug messages</td>
</tr>
<tr>
<td></td>
<td>Decides whether the received message is stored into an Rx buffer or treated</td>
</tr>
<tr>
<td><strong>Bits 10:9</strong></td>
<td><strong>SFID2[10:9]</strong></td>
</tr>
<tr>
<td></td>
<td>as message A, B, or C of the debug message sequence.</td>
</tr>
<tr>
<td></td>
<td>- 00: Store message into an Rx buffer</td>
</tr>
<tr>
<td></td>
<td>- 01: Debug message A</td>
</tr>
<tr>
<td></td>
<td>- 10: Debug message B</td>
</tr>
<tr>
<td></td>
<td>- 11: Debug message C</td>
</tr>
<tr>
<td><strong>SFID2[8:6]</strong></td>
<td>Is used to control the filter event pins at the Extension Interface. A 1 at</td>
</tr>
<tr>
<td></td>
<td>the respective bit position enables generation of a pulse at the related</td>
</tr>
<tr>
<td></td>
<td>filter event pin with the duration of one fdcan_pclk period in case the</td>
</tr>
<tr>
<td></td>
<td>filter matches. SFID2[8] is used by the calibration unit.</td>
</tr>
<tr>
<td><strong>SFID2[5:0]</strong></td>
<td>Defines the offset to the Rx buffer start address FDCAN_RXBC.RBSA for storage</td>
</tr>
<tr>
<td></td>
<td>of a matching message.</td>
</tr>
</tbody>
</table>

1. With SFT = “11” the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC = “000”).

**Note:** In case a reserved value is configured, the filter element is considered disabled.
61.4.22 FDCAN extended message ID filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended message ID filter element, its address is the filter list extended start address FDCAN_XIDFC.FLESA plus two times the index of the filter element (0 … 63).

### Table 483. Extended message ID filter element

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>F0</td>
<td>EFEC[2:0]</td>
<td>EFID1[28:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>EFTI[1:0]</td>
<td>Reserved</td>
<td>EFID2[28:0]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 484. Extended message ID filter element field description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| F0 bits 31:29 EFEC[2:0] | Extended filter element configuration  
All enabled filter elements are used for acceptance filtering of extended frames.  
Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = 100, 101, or 110 a match sets interrupt flag FDCAN_IR.HPM and, if enabled, an interrupt is generated. In this case register FDCAN_HPMS is updated with the status of the priority match.  
– 000: Disable filter element  
– 001: Store in Rx FIFO 0 if filter matches  
– 010: Store in Rx FIFO 1 if filter matches  
– 011: Reject ID if filter matches  
– 100: Set priority if filter matches  
– 101: Set priority and store in FIFO 0 if filter matches  
– 110: Set priority and store in FIFO 1 if filter matches  
– 111: Store into Rx buffer, configuration of EFT[1:0] ignored |
| F0 bits 28:0 EFID1[28:0] | Extended filter ID 1  
First ID of extended ID filter element.  
When filtering for Rx buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only FDCAN_XIDAM masking mechanism. |
| F1 bits 31:30 EFT[1:0] | Extended filter type  
– 00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID)  
– 01: Dual ID filter for EF1ID or EF2ID  
– 10: Classic filter: EF1ID = filter, EF2ID = mask  
– 11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), FDCAN_XIDAM mask not applied |
### 61.4.23 FDCAN trigger memory element

Up to 64 trigger memory elements can be configured. When accessing a trigger memory element, its address is the trigger memory start address FDCAN_TTTMC.TMSA plus the index of the trigger memory element (0 … 63).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>TM[15:0]</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>CC[6:0]</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>TMIN</td>
</tr>
<tr>
<td></td>
<td>TMEX</td>
</tr>
<tr>
<td></td>
<td>TYPE[3:0]</td>
</tr>
<tr>
<td>T1</td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>FTYPE</td>
</tr>
<tr>
<td></td>
<td>MNR[6:0]</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>MSC[2:0]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>Time mark</td>
</tr>
<tr>
<td></td>
<td>Cycle time for which the trigger becomes active.</td>
</tr>
<tr>
<td>T0 bit 14:8</td>
<td>Cycle code</td>
</tr>
<tr>
<td></td>
<td>0b0000000x valid for all cycles</td>
</tr>
<tr>
<td></td>
<td>0b0000001c valid every 2nd cycle at cycle count mod2 = c</td>
</tr>
<tr>
<td></td>
<td>0b000001cc valid every 4th cycle at cycle count mod4 = cc</td>
</tr>
<tr>
<td></td>
<td>0b0001cccc valid every 8th cycle at cycle count mod8 = ccc</td>
</tr>
<tr>
<td></td>
<td>0b0011cccc valid every 16th cycle at cycle count mod16 = ccccc</td>
</tr>
<tr>
<td></td>
<td>0b01cccccc valid every 32nd cycle at cycle count mod32 = cccccc</td>
</tr>
<tr>
<td></td>
<td>0b1ccccccc valid every 64th cycle at cycle count mod64 = ccccccc</td>
</tr>
</tbody>
</table>
Table 486. Trigger memory element description (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0 bit 5 TMIN</td>
<td>Time mark event internal</td>
</tr>
<tr>
<td></td>
<td>– 0: No action</td>
</tr>
<tr>
<td></td>
<td>– 1: FDCAN_TTIR.TTMI is set when trigger memory element becomes active</td>
</tr>
<tr>
<td>T0 bit 4 TMEX</td>
<td>Time mark event external</td>
</tr>
<tr>
<td></td>
<td>– 0: No action</td>
</tr>
<tr>
<td></td>
<td>– 1: Pulse at output fdcan1_tmp for more than one instance and fdcan_tmp if only one instance with the length of one period is generated when the time arc of the trigger memory element becomes active and FDCAN_TTOCN.TTMIE = 1</td>
</tr>
<tr>
<td>T0 bit 3:0 TYPE[3:0]</td>
<td>Trigger type</td>
</tr>
<tr>
<td></td>
<td>– 0000 Tx_Ref_Trigger - valid when not in Gap</td>
</tr>
<tr>
<td></td>
<td>– 0001 Tx_Ref_Trigger_Gap - valid when in Gap</td>
</tr>
<tr>
<td></td>
<td>– 0010 Tx_Trigger_Single - starts a single transmission in an exclusive time window</td>
</tr>
<tr>
<td></td>
<td>– 0011 Tx_Trigger_Continuous - starts continuous transmission in an exclusive time window</td>
</tr>
<tr>
<td></td>
<td>– 0100 Tx_Trigger_Arbitration - starts a transmission in an arbitrating time window</td>
</tr>
<tr>
<td></td>
<td>– 0101 Tx_Trigger_Merged - starts a merged arbitration window</td>
</tr>
<tr>
<td></td>
<td>– 0110 Watch_Trigger - valid when not in Gap</td>
</tr>
<tr>
<td></td>
<td>– 0111 Watch_Trigger_Gap - valid when in Gap</td>
</tr>
<tr>
<td></td>
<td>– 1000 Rx_Trigger - check for reception</td>
</tr>
<tr>
<td></td>
<td>– 1001 Time_Base_Trigger - only control TMIN, TMEX</td>
</tr>
<tr>
<td></td>
<td>– 1010 ... 1111=End_of_List - illegal type, causes configuration error</td>
</tr>
<tr>
<td>T1 bit 23FTYPE</td>
<td>Filter type</td>
</tr>
<tr>
<td></td>
<td>– 0: 11-bit standard message ID</td>
</tr>
<tr>
<td></td>
<td>– 1: 29-bit extended message ID</td>
</tr>
<tr>
<td>T1 bit 22:16 MNR[6:0][1]</td>
<td>Message Number</td>
</tr>
<tr>
<td></td>
<td>– Transmission: trigger is valid for configured Tx buffer number. Valid values are 0 to 31.</td>
</tr>
<tr>
<td></td>
<td>– Reception: trigger is valid for standard/extended message ID filter element number. Valid values are, respectively 0 to 63 and 0 to 127.</td>
</tr>
<tr>
<td>T1 bits 2:0 MSC[2:0]</td>
<td>Message status count</td>
</tr>
<tr>
<td></td>
<td>Counts scheduling errors for periodic messages in exclusive time windows. It has no function for arbitrating messages and in event-driven CAN communication (ISO11898-1).</td>
</tr>
<tr>
<td></td>
<td>– 0-7= Actual status</td>
</tr>
</tbody>
</table>

1. The trigger memory elements have to be written when the FDCAN is in INIT state. Write access to the trigger memory elements outside INIT state is not allowed. There is an exception for TMIN and TMEX when they are defined as part of a trigger memory element of TYPE Tx_Ref_Trigger. In this case they become active at the time mark modified by the actual reference trigger offset (TTOST[RTO]).
Controller area network with flexible data rate (FDCAN)

61.5 FDCAN registers

61.5.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000
Reset value: 0x3214 1218

<table>
<thead>
<tr>
<th>Bits</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:28</td>
<td>REL[3:0]: Core release = 3</td>
</tr>
<tr>
<td>27:24</td>
<td>STEP[3:0]: Step of core release = 2</td>
</tr>
<tr>
<td>23:20</td>
<td>SUBSTEP[3:0]: Sub-step of core release = 1</td>
</tr>
<tr>
<td>19:16</td>
<td>YEAR[3:0]: Timestamp Year = 4</td>
</tr>
<tr>
<td>15:8</td>
<td>MON[7:0]: Timestamp Month = 12</td>
</tr>
<tr>
<td>7:0</td>
<td>DAY[7:0]: Timestamp Day = 18</td>
</tr>
</tbody>
</table>

61.5.2 FDCAN Endian register (FDCAN_ENDN)

Address offset: 0x0004
Reset value: 0x8765 4321

<table>
<thead>
<tr>
<th>Bits</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>ETV[31:0]: Endianness Test value</td>
</tr>
</tbody>
</table>

The endianness test value is 0x8765 4321.

61.5.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C
Reset value: 0x0000 0A33

This register is dedicated to data bit timing phase and only writable if bits FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set. The CAN time quantum may be programmed in the range from 1 to 32 FDCAN clock periods. tq = (DBRP + 1) FDCAN clock periods.
DTSEG1 is the sum of Prop_Seg and Phase_Seg1. DTSEG2 is Phase_Seg2. Therefore the length of the bit time is (DTSEG1 + DTSEG2 + 3) tq for programmed values, or (Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2) tq for functional values.

The information processing time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Note: With a FDCAN clock of 8 MHz, the reset value 0x00000A33 configures the FDCAN for a fast bitrate of 500 kbit/s.

### 61.5.4 FDCAN test register (FDCAN_TEST)

Write access to this register has to be enabled by setting bit FDCAN_CCCR.TEST to 1. All register functions are set to their reset values when bit FDCAN_CCCR.TEST is reset.

Loop back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX differently from 00 may disturb the message transfer on the CAN bus.

Address offset: 0x0010
Reset value: 0x0000 0000
Controller area network with flexible data rate (FDCAN)

61.5.5 FDCAN RAM watchdog register (FDCAN_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured by the FDCAN_RWD.WDC bits.

The counter is reloaded with FDCAN_RWD.WDC bits when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to 0, the counter stops and interrupt flag FDCAN_IR.WDI bit is set. The RAM watchdog counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014
Reset value: 0x0000 0000

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

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

Bit 7 RX: Receive pin
- Monitors the actual value of transmit pin FDCANx_RX
  0: The CAN bus is dominant (FDCANx_RX = 0)
  1: The CAN bus is recessive (FDCANx_RX = 1)

Bits 6:5 TX[1:0]: Control of transmit pin
- 00: Reset value, FDCANx_TX TX is controlled by the CAN core, updated at the end of the CAN bit time
- 01: Sample point can be monitored at pin FDCANx_TX
- 10: Dominant (0) level at pin FDCANx_TX
- 11: Recessive (1) at pin FDCANx_TX

Bit 4 LBCK: Loop back mode
- 0: Reset value, loop back mode is disabled
- 1: Loop back mode is enabled (see Test modes)

Bits 3:0 Reserved, must be kept at reset value.
61.5.6 FDCAN CC control register (FDCAN_CCCR)

Address offset: 0x0018
Reset value: 0x0000 0001

For details about setting and resetting of single bits see Software initialization.

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NISO**: Non ISO operation
If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.
0: CAN FD frame format according to ISO11898-1
1: CAN FD frame format according to Bosch CAN FD Specification V1.0

Bit 14 **TXP**: 
If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.
0: disabled
1: enabled

Bit 13 **EFBI**: Edge Filtering during bus Integration
0: Edge filtering disabled
1: Two consecutive dominant tq required to detect an edge for hard synchronization

Bit 12 **PXHD**: Protocol exception handling disable
0: Protocol exception handling enabled
1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE**: FDCAN bitrate Switching
0: Bitrate switching for transmissions disabled
1: Bitrate switching for transmissions enabled
Bit 8 **FDOE**: FD operation enable
   0: FD operation disabled
   1: FD operation enabled

Bit 7 **TEST**: Test mode enable
   0: Normal operation, register TEST holds reset values
   1: Test mode, write access to register TEST enabled

Bit 6 **DAR**: Disable automatic retransmission
   0: Automatic retransmission of messages not transmitted successfully enabled
   1: Automatic retransmission disabled

Bit 5 **MON**: Bus monitoring mode
   Bit MON can only be set by software when both CCE and INIT are set to 1. The bit can be reset by the Host at any time.
   0: Bus monitoring mode is disabled
   1: Bus monitoring mode is enabled

Bit 4 **CSR**: Clock stop request
   0: No clock stop is requested
   1: Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle.

Bit 3 **CSA**: Clock stop acknowledge
   0: No clock stop acknowledged
   1: FDCAN may be set in power down by stopping APB clock and kernel clock

Bit 2 **ASM**: ASM restricted operation mode
   The restricted operation mode is intended for applications that adapt themselves to different CAN bitrates. The application tests different bitrates and leaves the restricted operation mode after it has received a valid frame. In the optional restricted operation mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by software when both CCE and INIT are set to 1. The bit can be reset by the software at any time. This bit will be set automatically set to 1 when the Tx handler was not able to read data from the message RAM in time.
   If the FDCAN is connected to a clock calibration on CAN unit, ASM bit is set by hardware as long as the calibration is not completed.
   0: Normal CAN operation
   1: Restricted operation mode active

Bit 1 **CCE**: Configuration change enable
   0: The CPU has no write access to the protected configuration registers
   1: The CPU has write access to the protected configuration registers (while FDCAN_CCCR.INIT = 1 CCE bit is automatically cleared when INIT bit is cleared)

Bit 0 **INIT**: Initialization
   0: Normal operation
   1: Initialization is started (while FDCAN_CCCR.INIT = 1 CCE bit is automatically cleared when INIT bit is cleared)

**Note:** Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to
assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

61.5.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C
Reset value: 0x06000A03

This register is dedicated to the nominal bit timing used during the arbitration phase, and is only writable if bits FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set. The CAN bit time may be programed in the range of 4 to 81 tq. The CAN time quantum may be programmed in the range of [1 … 1024] FDCAN kernel clock periods.

tq = \((BRP + 1) \cdot \text{FDCAN clock period} \cdot \text{fdcan_tq_ck}\)

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) \([\text{NTSEG1} + \text{NTSEG2} + 3] \cdot \text{tq}\) or (functional values) \([\text{Sync_Seg} + \text{Prop_Seg} + \text{Phase_Seg1} + \text{Phase_Seg2}] \cdot \text{tq}\).

The information processing time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

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

<table>
<thead>
<tr>
<th>15:8</th>
<th>0:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTSEG1[7:0]</td>
<td>NTSEG2[6:0]</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw</td>
<td>rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:25 **NSJW[6:0]:** Nominal (re)synchronization jump width
Should be smaller than NTSEG2, valid values are 0 to 127. The value used by the hardware is the one programmed, incremented by 1, i.e. \(t_{SJW} = (\text{NSJW} + 1) \cdot \text{tq}\).
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 24:16 **NBRP[8:0]:** Bitrate prescaler
Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values are 0 to 511. The value used by the hardware is the one programmed, incremented by 1.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:8 **NTSEG1[7:0]:** Nominal time segment before sample point
Valid values are 0 to 255. The value used by the hardware is the one programmed, incremented by 1, i.e. \(t_{BS1} = (\text{NTSEG1} + 1) \cdot \text{tq}\).
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 7 **Reserved, must be kept at reset value.**

Bits 6:0 **NTSEG2[6:0]:** Nominal time segment after sample point
Valid values are 0 to 127. The value used by the hardware is the one programmed, incremented by 1, i.e. \(t_{BS2} = (\text{NTSEG2} + 1) \cdot \text{tq}\).
Note: With a CAN kernel clock of 8 MHz, the reset value of 0x00000A33 configures the FDCAN for a bitrate of 125 kbit/s.

61.5.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020
Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TCP[3:0]**: Timestamp counter prescaler
Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 ... 16].
The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
In CAN FD mode the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS = 10).
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

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

Bits 1:0 **TSS[1:0]**: Timestamp select
00: Timestamp counter value always 0x0000
01: Timestamp counter value incremented according to TCP
10: External timestamp counter from TIM3 value used (tim3_cnt[0:15])
11: Same as 00.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

61.5.9 FDCAN timestamp counter value register (FDCAN_TSCV)

Address offset: 0x0024
Reset value: 0x0000 0000
Note: A "wrap around" is a change of the timestamp counter value from non-0 to 0 not caused by write access to FDCAN_TSCV.

### 61.5.10 FDCAN timeout counter configuration register (FDCAN_TOCC)

Address offset: 0x0028  
Reset value: 0xFFFF 0000
61.5.11 FDCAN timeout counter value register (FDCAN_TOCV)

Address offset: 0x002C
Reset value: 0x0000 FFFF

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

Bits 31:16 **TOP[15:0]**: Timeout period
Start value of the timeout counter (down-counter). Configures the timeout period.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select
When operating in Continuous mode, a write to FDCAN_TOCV presets the counter to the value configured by FDCAN_TOCC.TOP and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by FDCAN_TOCC.TOP. Down-counting is started when the first FIFO element is stored.

00: Continuous operation
01: Timeout controlled by Tx event FIFO
10: Timeout controlled by Rx FIFO 0
11: Timeout controlled by Rx FIFO 1

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 0 **ETOC**: Enable timeout counter

0: Timeout counter disabled
1: Timeout counter enabled

This is a protected write bit, write access is possible only when the bit 1 (CCE) and bit 0 (INIT) of FDCAN_CCCR register are set to 1.

For more details see **Timeout counter**.

61.5.12 FDCAN error counter register (FDCAN_ECR)

Address offset: 0x0040
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
r r r r r rc_r rc_r rc_r rc_r rc_r rc_r rc_r rc_r

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:16  CEL[7:0]: CAN error logging
The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag FDCAN_IR.ELO.

Bit 15  RP: Receive error passive
0: The receive error counter is below the error passive level of 128
1: The receive error counter has reached the error passive level of 128

Bits 14:8  REC[6:0]: Receive error counter
Actual state of the receive error counter, values between 0 and 127.

Bits 7:0  TEC[7:0]: Transmit error counter
Actual state of the transmit error counter, values between 0 and 255.
When FDCAN_CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

61.5.13  FDCAN protocol status register (FDCAN_PSR)
Address offset: 0x0044
Reset value: 0x0000 0707

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

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
r r r r r r r r r r

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

Bits 22:16  TDCV[6:0]: Transmitter Delay Compensation value
Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and FDCAN_TDCR.TDCO. The SSP position is, in the data phase, the number of minimum time quanta (mtq) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.

Bit 15  Reserved, must be kept at reset value.
Bit 14 **PXE**: Protocol exception event
- 0: No protocol exception event occurred since last read access
- 1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message
This bit is set independent of acceptance filtering.
- 0: Since this bit was reset by the CPU, no FDCAN message has been received
- 1: Message in FDCAN format with EDL flag set has been received
   Access type is RX: reset on read.

Bit 12 **RBRS**: BRS flag of last received FDCAN message
This bit is set together with REDL, independent of acceptance filtering.
- 0: Last received FDCAN message did not have its BRS flag set
- 1: Last received FDCAN message had its BRS flag set
   Access type is RX: reset on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message
This bit is set together with REDL, independent of acceptance filtering.
- 0: Last received FDCAN message did not have its ESI flag set
- 1: Last received FDCAN message had its ESI flag set
   Access type is RX: reset on read.

Bits 10:8 **DLEC[2:0]**: Data Last error code
Type of last error that occurred in the data phase of a FDCAN format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to 0 when a FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.
Access type is RS: set on read.

Bit 7 **BO**: Bus_Off status
- 0: The FDCAN is not Bus_Off
- 1: The FDCAN is in Bus_Off state

Bit 6 **EW**: Warning status
- 0: Both error counters are below the Error_Warning limit of 96
- 1: At least one of error counter has reached the Error_Warning limit of 96

Bit 5 **EP**: Error Passive
- 0: The FDCAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected
- 1: The FDCAN is in the Error_Passive state

Bits 4:3 **ACT[1:0]**: Activity
Monitors the module CAN communication state.
- 00: Synchronizing: node is synchronizing on CAN communication
- 01: Idle: node is neither receiver nor transmitter
- 10: Receiver: node is operating as receiver
- 11: Transmitter: node is operating as transmitter
Bits 2:0  **LEC[2:0]:** Last error code

The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to 0 when a message has been transferred (reception or transmission) without error.

- **000:** No error: No error occurred since LEC has been reset by successful reception or transmission.
- **001:** Stuff error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
- **010:** Form error: A fixed format part of a received frame has the wrong format.
- **011:** AckError: The message transmitted by the FDCAN was not acknowledged by another node.
- **100:** Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.
- **101:** Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
- **110:** CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.
- **111:** NoChange: Any read access to the protocol status register re-initializes the LEC to '7'. When the LEC shows the value 7, no CAN bus event was detected since the last CPU read access to the protocol status register.

Access type is RS: set on read.

**Note:** When a frame in FDCAN format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of a FDCAN CRC sequence will be shown as a Form error, not Stuff error.

**Note:** The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting FDCAN_CCCR.INIT. If the device goes Bus_Off, it will set FDCAN_CCCR.INIT of its own, stopping all bus activities. Once FDCAN_CCCR.INIT has been cleared by the CPU, the device will then wait for 129 occurrences of bus Idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the error management counters will be reset. During the waiting time after the reset of FDCAN_CCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0 error code is written to FDCAN_PSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. FDCAN_ECR.REC is used to count these sequences.

### 61.5.14 FDCAN transmitter delay compensation register (FDCAN_TDCR)

**Address offset:** 0x0048

**Reset value:** 0x0000 0000
61.5.15 FDCAN interrupt register (FDCAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position.

Writing a 0 has no effect. A hard reset will clear the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050
Reset value: 0x0000 0000

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

Bit 29 **ARA**: Access to Reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 28 **PED**: Protocol error in data phase (data Bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (PSR.DLEC different from 0,7)
Bit 27 **PEA**: Protocol error in Arbitration phase (Nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (PSR.LEC different from 0,7)

Bit 26 **WDI**: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY

Bit 25 **BO**: Bus_Off status
0: Bus_Off status unchanged
1: Bus_Off status changed

Bit 24 **EW**: Warning status
0: Error_Warning status unchanged
1: Error_Warning status changed

Bit 23 **EP**: Error Passive
0: Error_Passive status unchanged
1: Error_Passive status changed

Bit 22 **ELO**: Error logging Overflow
0: CAN error logging counter did not overflow
1: Overflow of CAN error logging counter occurred

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **DRX**: Message stored to dedicated Rx buffer
The flag is set whenever a received message has been stored into a dedicated Rx buffer.
0: No Rx buffer updated
1: At least one received message stored into a Rx buffer

Bit 18 **TOO**: Timeout occurred
0: No timeout
1: Timeout reached

Bit 17 **MRAF**: Message RAM Access Failure
The flag is set when the Rx handler
1. Has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
2. Was unable to write a message to the message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated or the New data flag for a dedicated Rx buffer is not set. The partly stored message is overwritten when the next message is stored to this location.

The flag is also set when the Tx handler was not able to read a message from the message RAM in time. In this case message transmission is aborted. In case of a Tx handler access failure the FDCAN is switched into restricted operation mode (see Restricted operation mode). To leave restricted operation mode, the Host CPU has to reset FDCAN_CCCR.ASM.

0: No message RAM access failure occurred
1: Message RAM access failure occurred
<table>
<thead>
<tr>
<th>Bit 16</th>
<th><strong>TSW</strong>: Timestamp wraparound</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No timestamp counter wraparound</td>
</tr>
<tr>
<td>1</td>
<td>Timestamp counter wraparound</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th><strong>TEFL</strong>: Tx event FIFO element Lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No Tx event FIFO element lost</td>
</tr>
<tr>
<td>1</td>
<td>Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size zero</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 14</th>
<th><strong>TEFF</strong>: Tx event FIFO Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tx event FIFO not full</td>
</tr>
<tr>
<td>1</td>
<td>Tx event FIFO full</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 13</th>
<th><strong>TEFW</strong>: Tx event FIFO Watermark Reached</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tx event FIFO fill level below watermark</td>
</tr>
<tr>
<td>1</td>
<td>Tx event FIFO fill level reached watermark</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 12</th>
<th><strong>TEFN</strong>: Tx event FIFO New Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tx event FIFO unchanged</td>
</tr>
<tr>
<td>1</td>
<td>Tx handler wrote Tx event FIFO element</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 11</th>
<th><strong>TFE</strong>: Tx FIFO Empty</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tx FIFO non-empty</td>
</tr>
<tr>
<td>1</td>
<td>Tx FIFO empty</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 10</th>
<th><strong>TCF</strong>: Transmission cancellation finished</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No transmission cancellation finished</td>
</tr>
<tr>
<td>1</td>
<td>Transmission cancellation finished</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 9</th>
<th><strong>TC</strong>: Transmission Completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No transmission completed</td>
</tr>
<tr>
<td>1</td>
<td>Transmission completed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 8</th>
<th><strong>HPM</strong>: High priority message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No high priority message received</td>
</tr>
<tr>
<td>1</td>
<td>High priority message received</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 7</th>
<th><strong>RF1L</strong>: Rx FIFO 1 message Lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No Rx FIFO 1 message lost</td>
</tr>
<tr>
<td>1</td>
<td>Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 6</th>
<th><strong>RF1F</strong>: Rx FIFO 1 Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Rx FIFO 1 not full</td>
</tr>
<tr>
<td>1</td>
<td>Rx FIFO 1 full</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 5</th>
<th><strong>RF1W</strong>: Rx FIFO 1 Watermark Reached</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Rx FIFO 1 fill level below watermark</td>
</tr>
<tr>
<td>1</td>
<td>Rx FIFO 1 fill level reached watermark</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 4</th>
<th><strong>RF1N</strong>: Rx FIFO 1 new message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No new message written to Rx FIFO 1</td>
</tr>
<tr>
<td>1</td>
<td>New message written to Rx FIFO 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 3</th>
<th><strong>RF0L</strong>: Rx FIFO 0 message Lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No Rx FIFO 0 message lost</td>
</tr>
<tr>
<td>1</td>
<td>Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero</td>
</tr>
</tbody>
</table>
Bit 2 **RF0F**: Rx FIFO 0 Full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bit 1 **RF0W**: Rx FIFO 0 Watermark Reached
0: Rx FIFO 0 fill level below watermark
1: Rx FIFO 0 fill level reached watermark

Bit 0 **RF0N**: Rx FIFO 0 New message
0: No new message written to Rx FIFO 0
1: New message written to Rx FIFO 0

### 61.5.16 FDCAN interrupt enable register (FDCAN_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register will be signaled on an interrupt line.

**Address offset**: 0x0054

**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>ARAE</td>
<td>Access to Reserved address enable</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>PEDE</td>
<td>Protocol error in data phase enable</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>PEAE</td>
<td>Protocol error in Arbitration phase enable</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>WDIE</td>
<td>Watchdog interrupt enable 0: Interrupt disabled 1: Interrupt enabled</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>BOE</td>
<td>Bus_Off status 0: Interrupt disabled 1: Interrupt enabled</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>EWE</td>
<td>Warning status interrupt enable 0: Interrupt disabled 1: Interrupt enabled</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>EPE</td>
<td>Error passive interrupt enable 0: Interrupt disabled 1: Interrupt enabled</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>ELOE</td>
<td>Error logging overflow interrupt enable 0: Interrupt disabled 1: Interrupt enabled</td>
<td></td>
</tr>
</tbody>
</table>

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

- **Bit 29 ARAE**: Access to Reserved address enable
- **Bit 28 PEDE**: Protocol error in data phase enable
- **Bit 27 PEAE**: Protocol error in Arbitration phase enable
- **Bit 26 WDIE**: Watchdog interrupt enable
  - 0: Interrupt disabled
  - 1: Interrupt enabled
- **Bit 25 BOE**: Bus_Off status
  - 0: Interrupt disabled
  - 1: Interrupt enabled
- **Bit 24 EWE**: Warning status interrupt enable
  - 0: Interrupt disabled
  - 1: Interrupt enabled
- **Bit 23 EPE**: Error passive interrupt enable
  - 0: Interrupt disabled
  - 1: Interrupt enabled
- **Bit 22 ELOE**: Error logging overflow interrupt enable
  - 0: Interrupt disabled
  - 1: Interrupt enabled
Bits 21:20  Reserved, must be kept at reset value.

Bit 19  DRXE: Message stored to dedicated Rx buffer interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled

Bit 18  TOOE: Timeout occurred interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled

Bit 17  MRAFE: Message RAM access failure interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled

Bit 16  TSWE: Timestamp wraparound interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled

Bit 15  TEFLE: Tx event FIFO element lost interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled

Bit 14  TEFFE: Tx event FIFO full interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled

Bit 13  TEFWE: Tx event FIFO watermark reached interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled

Bit 12  TEFNE: Tx event FIFO new entry interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled

Bit 11  TFEE: Tx FIFO empty interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled

Bit 10  TCFE: Transmission cancellation finished interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled

Bit 9   TCE: Transmission completed interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled

Bit 8   HPME: High priority message interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled

Bit 7   RF1LE: Rx FIFO 1 message lost interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled

Bit 6   RF1FE: Rx FIFO 1 full interrupt enable
  0: Interrupt disabled
  1: Interrupt enabled
61.5.17 FDCAN interrupt line select register (FDCAN_ICS)

This register assigns an interrupt generated by a specific interrupt flag from the interrupt register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via FDCAN_IIE.EINT0 and FDCAN_IIE.EINT1.

Address offset: 0x0058
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>Res.</td>
<td>ARAL</td>
<td>PEDL</td>
<td>PEAL</td>
<td>WDIL</td>
<td>BOL</td>
<td>EWL</td>
<td>EPL</td>
<td>ELOL</td>
<td>DRXL</td>
<td>TOOL</td>
<td>MRAFL</td>
<td>TSWL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TEFLL</td>
<td>TEFRL</td>
<td>TEFWL</td>
<td>TEFNL</td>
<td>TFEL</td>
<td>TCFL</td>
<td>TCL</td>
<td>HPML</td>
<td>RF1LL</td>
<td>RF1FL</td>
<td>RF1WL</td>
<td>RF1NL</td>
<td>RF0LL</td>
<td>RF0FL</td>
<td>RF0WL</td>
<td>RF0NL</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

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

- Bit 29 ARAL: Access to reserved address line
- Bit 28 PEDL: Protocol error in data phase line
- Bit 27 PEAL: Protocol error in arbitration phase line
- Bit 26 WDIL: Watchdog interrupt Line
- Bit 25 BOL: Bus_Off status
- Bit 24 EWL: Warning status interrupt Line
- Bit 23 EPL: Error passive interrupt line
Bit 22  **ELOL**: Error logging overflow interrupt line

Bits 21:20  Reserved, must be kept at reset value.

Bit 19  **DRXL**: Message stored to dedicated Rx buffer interrupt line

Bit 18  **TOOL**: Timeout occurred interrupt Line

Bit 17  **MRAFL**: Message RAM access failure interrupt line

Bit 16  **TSWL**: Timestamp wraparound interrupt line

Bit 15  **TEFLL**: Tx event FIFO element Lost interrupt line

Bit 14  **TEFFL**: Tx event FIFO full interrupt line

Bit 13  **TEFWL**: Tx event FIFO watermark reached interrupt line

Bit 12  **TEFNL**: Tx event FIFO new entry interrupt line

Bit 11  **TFEL**: Tx FIFO empty interrupt Line

Bit 10  **TCFL**: Transmission cancellation finished interrupt line

Bit 9  **TCL**: Transmission completed interrupt line

Bit 8  **HPML**: High priority message interrupt line

Bit 7  **RF1LL**: Rx FIFO 1 message lost interrupt line

Bit 6  **RF1FL**: Rx FIFO 1 full interrupt line

Bit 5  **RF1WL**: Rx FIFO 1 watermark reached interrupt line

Bit 4  **RF1NL**: Rx FIFO 1 new message interrupt line

Bit 3  **RF0LL**: Rx FIFO 0 message lost interrupt line

Bit 2  **RF0FL**: Rx FIFO 0 full interrupt line

Bit 1  **RF0WL**: Rx FIFO 0 watermark reached interrupt line

Bit 0  **RF0NL**: Rx FIFO 0 new message interrupt line

### 61.5.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming bits EINT0 and EINT1.

Address offset: 0x005C

Reset value: 0x0000 0000

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

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

<table>
<thead>
<tr>
<th></th>
<th>EINT1</th>
<th>EINT0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
61.5.19  FDCAN global filter configuration register (FDCAN_GFC)

Global settings for message ID filtering. The global filter configuration register controls the filter path for standard and extended messages as described in Figure 704: Standard message ID filter path and Figure 705: Extended message ID filter path.

Address offset: 0x0080
Reset value: 0x0000 0000

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

Bits 31:6  Reserved, must be kept at reset value.

Bits 5:4  ANFS[1:0]: Accept non-matching frames standard
Defines how received messages with 11-bit ID that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 3:2  ANFE[1:0]: Accept non-matching frames extended
Defines how received messages with 29-bit ID that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1  EINT1: Enable interrupt Line 1
0: Interrupt line fdcan_intr1_it disabled
1: Interrupt line fdcan_intr1_it enabled

Bits 0  EINT0: Enable interrupt Line 0
0: Interrupt line fdcan_intr0_it disabled
1: Interrupt line fdcan_intr0_it enabled
Bit 1 **RRFS**: Reject remote frames standard
0: Filter remote frames with 11-bit standard ID
1: Reject all remote frames with 11-bit standard ID
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 0 **RRFE**: Reject remote frames extended
0: Filter remote frames with 29-bit standard ID
1: Reject all remote frames with 29-bit standard ID
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

### 61.5.20 FDCAN standard ID filter configuration register (FDCAN_SIDFC)

Settings for 11-bit standard message ID filtering. The standard ID filter configuration register controls the filter path for standard messages as described in Figure 704.

Address offset: 0x0084
Reset value: 0x0000 0000

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

Bits 31:24: Reserved, must be kept at reset value.

Bits 23:16 **LSS[7:0]**: List size standard
0: No standard message ID filter
1-128: Number of standard message ID filter elements
>128: Values greater than 128 are interpreted as 128.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **FLSSA[13:0]**: Filter list standard start address
Start address of standard message ID filter list (32-bit word address, see Table 481: Standard message ID filter element). These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

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

### 61.5.21 FDCAN extended ID filter configuration register (FDCAN_XIDFC)

Settings for 29-bit extended filter configuration. The FDCAN extended ID filter configuration register controls the filter path for standard messages as described in Figure 705: Extended message ID filter path.

Address offset: 0x0088
Reset value: 0x0000 0000
61.5.22 FDCAN extended ID and mask register (FDCAN_XIDAM)

Address offset: 0x0090
Reset value: 0x1FFF FFFF

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **LSE[7:0]**: List size extended
0: No extended message ID filter
1-128: Number of extended ID filter elements
>128: Values greater than 128 are interpreted as 128.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **FLESA[13:0]**: Filter list extended start address
Start address of extended message ID filter list (32-bit word address, see Table 483: Extended message ID filter element).
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **EIDM[28:0]**: Extended ID Mask
For acceptance filtering of extended frames the extended ID AND Mask is AND-ed with the message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939.
With the reset value of all bits set to 1 the mask is not active.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.
61.5.23  FDCAN high priority message status register (FDCAN_HPMS)

This register is updated every time a message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0094
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>FLST</th>
<th>FIDX[6:0]</th>
<th>MSI[1:0]</th>
<th>BIDX[5:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>30</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>29</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>28</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>27</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>26</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>25</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>24</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>23</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>22</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>21</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>20</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>18</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>17</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>16</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bit 15  FLST: Filter list
Indicates the filter list of the matching filter element.
0: Standard filter list
1: Extended filter list

Bits 14:8  FIDX[6:0]: Filter index
Index of matching filter element. Range is 0 to FDCAN_SIDFC[LSS] - 1 or FDCAN_XIDFC[LSE] - 1.

Bits 7:6  MSI[1:0]: Message Storage Indicator
00: No FIFO selected
01: FIFO overrun
10: Message stored in FIFO 0
11: Message stored in FIFO 1

Bits 5:0  BIDX[5:0]: Buffer index
Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

61.5.24  FDCAN new data 1 register (FDCAN_NDAT1)

Address offset: 0x0098
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>ND31</th>
<th>ND30</th>
<th>ND29</th>
<th>ND28</th>
<th>ND27</th>
<th>ND26</th>
<th>ND25</th>
<th>ND24</th>
<th>ND23</th>
<th>ND22</th>
<th>ND21</th>
<th>ND20</th>
<th>ND19</th>
<th>ND18</th>
<th>ND17</th>
<th>ND16</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>ND15</td>
<td>ND14</td>
<td>ND13</td>
<td>ND12</td>
<td>ND11</td>
<td>ND10</td>
<td>ND9</td>
<td>ND8</td>
<td>ND7</td>
<td>ND6</td>
<td>ND5</td>
<td>ND4</td>
<td>ND3</td>
<td>ND2</td>
<td>ND1</td>
<td>ND0</td>
</tr>
<tr>
<td>28</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
61.5.25  FDCAN new data 2 register (FDCAN_NDAT2)

Address offset: 0x009C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>ND[31:0]: New data[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The register holds the New data flags of Rx buffers 0 to 31. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register.</td>
</tr>
<tr>
<td></td>
<td>0: Rx buffer not updated</td>
</tr>
<tr>
<td></td>
<td>1: Rx buffer updated from new message</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address offset: 0x009C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value: 0x0000 0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>ND63</td>
<td>ND62</td>
<td>ND61</td>
<td>ND60</td>
<td>ND59</td>
<td>ND58</td>
<td>ND57</td>
<td>ND56</td>
<td>ND55</td>
<td>ND54</td>
<td>ND53</td>
<td>ND52</td>
<td>ND51</td>
<td>ND50</td>
<td>ND49</td>
<td>ND48</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>ND[63:32]: New data[63:32]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The register holds the New data flags of Rx buffers 32 to 63. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register.</td>
</tr>
<tr>
<td></td>
<td>0: Rx buffer not updated</td>
</tr>
<tr>
<td></td>
<td>1: Rx buffer updated from new message</td>
</tr>
</tbody>
</table>

61.5.26  FDCAN Rx FIFO 0 configuration register (FDCAN_RXF0C)

Address offset: 0x00A0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>F0OM</td>
<td>F0WM[6:0]</td>
<td>Res.</td>
<td>F0S[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| F0SA[13:0] |
| Res. | Res. |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
Bit 31 **F0OM**: FIFO 0 operation mode
- FIFO 0 can be operated in blocking or in overwrite mode.
  - 0: FIFO 0 blocking mode
  - 1: FIFO 0 overwrite mode

Bits 30:24 **F0WM[6:0]**: FIFO 0 Watermark
- 0: Watermark interrupt disabled
- 1-64: Level for Rx FIFO 0 watermark interrupt (FDCAN_IR.RF0W)
- >64: Watermark interrupt disabled
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **F0S[6:0]**: Rx FIFO 0 size
- 0: No Rx FIFO 0
- 1-64: Number of Rx FIFO 0 elements
- >64: Values greater than 64 are interpreted as 64
The Rx FIFO 0 elements are indexed from 0 to F0S-1.

Bits 15:2 **F0SA[13:0]**: Rx FIFO 0 start address
Start address of Rx FIFO 0 in message RAM (32-bit word address, see Figure 703).

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

### 61.5.27 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)

Address offset: 0x00A4
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Res</td>
<td>F0GI[5:0]</td>
<td>Res</td>
<td>F0FL[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:26 Reserved, must be kept at reset value.

- **Bit 25 RF0L**: Rx FIFO 0 message Lost
  - This bit is a copy of interrupt flag FDCAN_IR.RF0L. When FDCAN_IR.RF0L is reset, this bit is also reset.
  - 0: No Rx FIFO 0 message lost
  - 1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero

- **Bit 24 F0F**: Rx FIFO 0 Full
  - 0: Rx FIFO 0 not full
  - 1: Rx FIFO 0 full

Bits 23:22 Reserved, must be kept at reset value.

- **Bits 21:16 F0PI[5:0]**: Rx FIFO 0 put index
  - Rx FIFO 0 write index pointer, range 0 to 63.
61.5.28 **FDCAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)**

Address offset: 0x00A8

Reset value: 0x0000 0000

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

Bits 15:14  Reserved, must be kept at reset value.

Bits 13:8  **F0GI[5:0]**: Rx FIFO 0 get index

Rx FIFO 0 read index pointer, range 0 to 63.

Bit 7  Reserved, must be kept at reset value.

Bits 6:0  **F0FL[6:0]**: Rx FIFO 0 fill level

Number of elements stored in Rx FIFO 0, range 0 to 64.

61.5.29 **FDCAN Rx buffer configuration register (FDCAN_RXBC)**

Address offset: 0x00AC

Reset value: 0x0000 0000

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

Bits 31:6  Reserved, must be kept at reset value.

Bits 5:0  **F0AI[5:0]**: Rx FIFO 0 acknowledge index

After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 get index FDCAN_RXF0S.F0GI to F0AI + 1 and update the FIFO 0 fill level FDCAN_RXF0S.F0FL.
**61.5.30 FDCAN Rx FIFO 1 configuration register (FDCAN_RXF1C)**

Address offset: 0x00B0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1OM</td>
<td>F1WM[6:0]</td>
<td>Res.</td>
<td>F1S[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

**Bit 31**  
**F1OM**: FIFO 1 Operation mode  
FIFO 1 can be operated in blocking or in overwrite mode.  
0: FIFO 1 blocking mode  
1: FIFO 1 overwrite mode

**Bits 30:24**  
**F1WM[6:0]**: Rx FIFO 1 Watermark  
0: Watermark interrupt disabled  
1-64: Level for Rx FIFO 1 watermark interrupt (FDCAN_IR.RF1W)  
>64: Watermark interrupt disabled.  
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

**Bit 23** Reserved, must be kept at reset value.

**Bits 22:16**  
**F1S[6:0]**: Rx FIFO 1 size  
0: No Rx FIFO 1  
1-64: Number of Rx FIFO 1 elements  
>64: Values greater than 64 are interpreted as 64  
The Rx FIFO 1 elements are indexed from 0 to F1S - 1.  
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

**Bits 15:2**  
**F1SA[13:0]**: Rx FIFO 1 start address  
start address of Rx FIFO 1 in message RAM (32-bit word address, see Figure 703: Message RAM configuration).  
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

**Bits 1:0** Reserved, must be kept at reset value.
61.5.31 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x00B4
Reset value: 0x0000 0000

Bits 31:30 **DMS[1:0]**: Debug message status
- 00: Idle state, wait for reception of debug messages
- 01: Debug message A received
- 10: Debug messages A, B received
- 11: Debug messages A, B, C received

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **RF1L**: Rx FIFO 1 message Lost
This bit is a copy of interrupt flag FDCAN_IR.RF1L. When FDCAN_IR.RF1L is reset, this bit is also reset.
- 0: No Rx FIFO 1 message lost
- 1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero.

Bit 24 **F1F**: Rx FIFO 1 Full
- 0: Rx FIFO 1 not full
- 1: Rx FIFO 1 full

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **F1PI[5:0]**: Rx FIFO 1 put index
Rx FIFO 1 write index pointer, range 0 to 63.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **F1GI[5:0]**: Rx FIFO 1 get index
Rx FIFO 1 read index pointer, range 0 to 63.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **F1FL[6:0]**: Rx FIFO 1 fill level
Number of elements stored in Rx FIFO 1, range 0 to 64
61.5.32 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x00B8
Reset value: 0x0000 0000

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

Bits 31:6 Reserved, must be kept at reset value.
Bits 5:0 F1AI[5:0]: Rx FIFO 1 acknowledge index

F1AI[5:0]: Rx FIFO 1 acknowledge index
After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 get index FDCAN_RXF1S.F1GI to F1AI + 1 and update the FIFO 1 fill level FDCAN_RXF1S.F1FL.

61.5.33 FDCAN Rx buffer element size configuration register (FDCAN_RXESC)

Configures the number of data bytes belonging to an Rx buffer / Rx FIFO element. Data field sizes higher than 8 bytes are intended for CAN FD operation only.
Address offset: 0x00BC
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.
Bits 10:8 RBDS[2:0]: Rx buffer data Field size:
000: 8 byte data field
001: 12 byte data field
010: 16 byte data field
011: 20 byte data field
100: 24 byte data field
101: 32 byte data field
110: 48 byte data field
111: 64 byte data field
61.5.34 FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0
Reset value: 0x0000 0000

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 TFQM: Tx FIFO/queue mode.
0: Tx FIFO operation
1: Tx queue operation.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 29:24 TFQS[5:0]: Transmit FIFO/queue size.
0: No Tx FIFO/queue
1-32: Number of Tx buffers used for Tx FIFO/queue
>32: Values greater than 32 are interpreted as 32.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.
Note: The sum of TFQS and NDTB cannot be larger than 32. There is no check for erroneous configurations. The Tx buffers section in the message RAM starts with the dedicated Tx buffers.

### 61.5.35 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in register FDCAN_TXBRP. Therefore the effect of add/cancellation requests may be delayed due to a running Tx scan (FDCAN_TXBRP not yet updated).

**Address offset:** 0x00C4

**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:22</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 21</td>
<td><strong>TFQF</strong>: Tx FIFO/queue Full</td>
</tr>
<tr>
<td>0</td>
<td>Tx FIFO/queue not full</td>
</tr>
<tr>
<td>1</td>
<td>Tx FIFO/queue full</td>
</tr>
<tr>
<td>Bit 20:16</td>
<td><strong>TFQPI[4:0]</strong>: Tx FIFO/queue put index</td>
</tr>
<tr>
<td></td>
<td>Tx FIFO/queue write index pointer, range 0 to 31</td>
</tr>
<tr>
<td>Bit 15:13</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 12:8</td>
<td><strong>TFGI[4:0]</strong>: Tx FIFO get index</td>
</tr>
<tr>
<td></td>
<td>Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx queue operation is configured (FDCAN_TXBC.TFQM = 1)</td>
</tr>
</tbody>
</table>
Bits 7:6  Reserved, must be kept at reset value.

Bits 5:0  **TFFL[5:0]**: Tx FIFO free level  
Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx queue operation is configured (FDCAN_TXBC.TFQM = 1).

**Note:** In case of mixed configurations where dedicated Tx buffers are combined with a Tx FIFO or a Tx queue, the put and get index indicate the number of the Tx buffer starting with the first dedicated Tx buffers. For example: For a configuration of 12 dedicated Tx buffers and a Tx FIFO of 20 buffers a put index of 15 points to the fourth buffer of the Tx FIFO.

### 61.5.36  FDCAN Tx buffer element size configuration register  
**(FDCAN_TXESC)**

Configures the number of data bytes belonging to a Tx buffer element. Data field sizes >8 bytes are intended for CAN FD operation only.

Address offset: 0x00C8  
Reset value: 0x0000 0000

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

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

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

Bits 2:0  **TBDS[2:0]**: Tx buffer data Field size:  
000: 8 byte data field  
001: 12 byte data field  
010: 16 byte data field  
011: 20 byte data field  
100: 24 byte data field  
101: 32 byte data field  
110: 48 byte data field  
111: 64 byte data field
61.5.37  **FDCAN Tx buffer request pending register (FDCAN_TXBRP)**

Address offset: 0x00CC
Reset value: 0x0000 0000

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

| 15:0 |
| TRP[15:0] |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

**Bits 31:0  TRP[31:0]: Transmission request pending.**

- Each Tx buffer has its own transmission request pending bit. The bits are set via register FDCAN_TXBAR. The bits are reset after a requested transmission has completed or has been canceled via register FDCAN_TXBCR.
- FDCAN_TXBRP bits are set only for those Tx buffers configured via FDCAN_TXBC. After a FDCAN_TXBRP bit has been set, a Tx scan (see **Filtering for Debug messages**) is started to check for the pending Tx request with the highest priority (Tx buffer with lowest message ID).
- A cancellation request resets the corresponding transmission request pending bit of register FDCAN_TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding FDCAN_TXBRP bit has been reset.
- After a cancellation has been requested, a finished cancellation is signaled via FDCAN_TXBCF
  - after successful transmission together with the corresponding FDCAN_TXBTO bit
  - when the transmission has not yet been started at the point of cancellation
  - when the transmission has been aborted due to lost arbitration
  - when an error occurred during frame transmission
- In DAR mode all transmissions are automatically canceled if they are not successful. The corresponding FDCAN_TXBCF bit is set for all unsuccessful transmissions.
- 0: No transmission request pending
- 1: Transmission request pending

**Note:**  
FDCAN_TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is canceled immediately, the corresponding FDCAN_TXBRP bit is reset.
61.5.38 FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00D0
Reset value: 0x0000 0000

Bits 31:0 AR[31:0]: Add request
Each Tx buffer has its own add request bit. Writing a 1 will set the corresponding add request bit; writing a 0 has no impact. This enables the Host to set transmission requests for multiple Tx buffers with one write to FDCAN_TXBAR. FDCAN_TXBAR bits are set only for those Tx buffers configured via FDCAN_TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.
0: No transmission request added
1: Transmission requested added.

Note: If an add request is applied for a Tx buffer with pending transmission request (corresponding FDCAN_TXBRP bit already set), the request is ignored.

61.5.39 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D4
Reset value: 0x0000 0000

Bits 31:0 CR[31:0]: Cancellation request
Each Tx buffer has its own cancellation request bit. Writing a 1 will set the corresponding cancellation request bit; writing a 0 has no impact. This enables the Host to set cancellation requests for multiple Tx buffers with one write to FDCAN_TXBCR. FDCAN_TXBCR bits are set only for those Tx buffers configured via FDCAN_TXBC. The bits remain set until the corresponding FDCAN_TXBRP bit is reset.
0: No cancellation pending
1: Cancellation pending
61.5.40  FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **TO[31:0]**: Transmission occurred.
Each Tx buffer has its own transmission occurred bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmission occurred
1: Transmission occurred

61.5.41  FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00DC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **CF[31:0]**: Cancellation finished
Each Tx buffer has its own cancellation finished bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a cancellation was requested via FDCAN_TXBCR. In case the corresponding FDCAN_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmit buffer cancellation
1: Transmit buffer cancellation finished

61.5.42  FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE)

Address offset: 0x00E0
Reset value: 0x0000 0000
### 61.5.43 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE)

**Address offset:** 0x00E4  
**Reset value:** 0x0000 0000

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

**Bits 31:0**  
**CFIE[31:16]:** Cancellation finished interrupt enable.  
Each Tx buffer has its own cancellation finished interrupt enable bit.  
0: Cancellation finished interrupt disabled  
1: Cancellation finished interrupt enabled

### 61.5.44 FDCAN Tx event FIFO configuration register (FDCAN_TXEFC)

**Address offset:** 0x00F0  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
61.5.45 FDCAN Tx event FIFO status register (FDCAN_TXEFS)

Address offset: 0x00F4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TEFL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EFS[5:0]</td>
</tr>
</tbody>
</table>

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TEFL**: Tx event FIFO element Lost.
This bit is a copy of interrupt flag FDCAN_IR.TEFL. When FDCAN_IR.TEFL is reset, this
bit is also reset.
0 No Tx event FIFO element lost
1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size zero.

Bit 24 **EFF**: Event FIFO Full.
0: Tx event FIFO not full
1: Tx event FIFO full

Bits 23:21 Reserved, must be kept at reset value.
Bits 20:16 **EFPI[4:0]**: Event FIFO put index.
   Tx event FIFO write index pointer, range 0 to 31.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **EFGI[4:0]**: Event FIFO get index.
   Tx event FIFO read index pointer, range 0 to 31.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **EFFL[5:0]**: Event FIFO fill level.
   Number of elements stored in Tx event FIFO, range 0 to 31.

### 61.5.46 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)

Address offset: 0x00F8

Reset value: 0x0000 0000

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

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **EFAI[4:0]**: Event FIFO acknowledge index.
   After the Host has read an element or a sequence of elements from the Tx event FIFO, it has to write the index of the last element read from Tx event FIFO to EFAI. This will set the Tx event FIFO get index FDCAN_TXEFS.EFGI to EFAI + 1 and update the FIFO fill level FDCAN_TXEFS.EFFL.

### 61.5.47 FDCAN TT trigger memory configuration register (FDCAN_TTTMC)

Address offset: 0x100

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMSA[13:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

61.5.48  **FDCAN TT reference message configuration register**  
* (FDCAN_TTRMC)  
Address offset: 0x0104  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>RMPS</th>
<th>XTD</th>
<th>Res.</th>
<th>RID[28:16]</th>
<th>RID[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  **RMPS**: Reference message Payload select  
Ignored in case of time slaves.  
0: Reference message has no additional payload  
1: The following elements are taken from Tx buffer 0:  
Message marker MM,  
Event FIFO control EFC,  
Data length code DLC,  
Data Bytes DB (level 1: bytes 2-8, level 0, 2: bytes 5-8)  
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 30  **XTD**: Extended identifier  
0: 11-bit standard identifier  
1: 29-bit extended identifier  
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.
Bit 29  Reserved, must be kept at reset value.

Bits 28:0  **RID[28:0]**: Reference identifier.
Identifier transmitted with reference message and used for reference message filtering.
Standard or extended reference identifier depending on bit XTD. A standard identifier has
to be written to ID[28:18].
These are protected write bits, write access is possible only when bit CCE and bit INIT of
FDCAN_CCCR register are set to 1.

### 61.5.49  FDCAN TT operation configuration register (FDCAN_TTOCF)

Address offset: 0x0108
Reset value: 0x0001 0000

<table>
<thead>
<tr>
<th>Bit 31:27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23:16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>EVTP</td>
<td>ECC</td>
<td>EGTF</td>
<td>AWL[7:0]</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15 14 13</td>
<td>12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 26  **EVTP**: Event trigger polarity.
0: Rising edge trigger
1: Falling edge trigger
These are protected write bits, write access is possible only when bit CCE and bit INIT of
FDCAN_CCCR register are set to 1.

Bit 25  **ECC**: Enable clock calibration.
0: Automatic clock calibration in FDCAN level 0, 2 is disabled
1: Automatic clock calibration in FDCAN level 0, 2 is enabled
These are protected write bits, write access is possible only when bit CCE and bit INIT of
FDCAN_CCCR register are set to 1.

Bit 24  **EGTF**: Enable global time Filtering.
0: Global time filtering in FDCAN level 0, 2 is disabled
1: Global time filtering in FDCAN level 0, 2 is enabled
These are protected write bits, write access is possible only when bit CCE and bit INIT of
FDCAN_CCCR register are set to 1.

Bits 23:16  **AWL[7:0]**: Application watchdog limit.
The application watchdog can be disabled by programming AWL to 0x00.
0x00–FF: Maximum time after which the application has to serve the application
watchdog. The application watchdog is incremented once each 256 NTUs.
These are protected write bits, write access is possible only when bit CCE and bit INIT of
FDCAN_CCCR register are set to 1.
Bit 15 **EECS**: Enable external clock synchronization
If enabled, TUR configuration (FDCAN_TURCF.NCL only) may be updated during FDCAN operation.
0: External clock synchronization in FDCAN level 0, 2 disabled
1: External clock synchronization in FDCAN level 0, 2 enabled
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 14:8 **IRTO[6:0]**: Initial reference trigger offset.
0x00–7F Positive offset, range from 0 to 127
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 7:5 **LDSDL[2:0]**: LD of synchronization deviation limit.
The synchronization deviation limit SDL is configured by its dual logarithm LDSDL with
SDL = 2 * (LDSDL + 5). SDL is comprised between 32 and 4096. It should not exceed the clock tolerance given by the CAN bit timing configuration.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 4 **TM**: Time master.
0: Time master function disabled
1: Potential time master
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 3 **GEN**: Gap enable.
0: Strictly time-triggered operation
1: External event-synchronized time-triggered operation
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **OM[1:0]**: Operation mode.
00: Event-driven CAN communication, default
01: TTCAN level 1
10: TTCAN level 2
11: TTCAN level 0
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

### 61.5.50 FDCAN TT matrix limits register (FDCAN_TTMLM)
Address offset: 0x010C
Reset value: 0x0000 0000
Note: ISO 11898-4, Section 5.2.1 requires that only the listed cycle count values are configured. Other values are possible, but may lead to inconsistent matrix cycles.

61.5.51 FDCAN TUR configuration register (FDCAN_TURCF)

The length of the NTU is given by: \( \text{NTU} = \text{CAN clock period} \times \text{NC}/\text{DC} \).

NC is an 18-bit value. Its high part, \( \text{NCH}[17:16] \) is hard wired to 0b01. Therefore the range of NC extends from 0x1000 to 0x1FFFF. The value configured by NCL is the initial value for...
FDCAN_TURNA/nav[15:0]. DC is set to 0x1000 by hardware reset and it may not be written to 0x0000.
- Level 1: NC 4 × DC and NTU = CAN bit time
- Levels 0 and 2: NC 8 × DC

The actual value of FDCAN_TUR may be changed by the clock drift compensation function of TTCAN level 0 and level 2 in order to adjust the node local view of the NTU to the time master view of the NTU. DC will not be changed by the automatic drift compensation, FDCAN_TURNA/nav may be adjusted around NC in the range of the synchronization deviation limit given by FDCAN_TTOCF.LDSDL. NC and DC should be programmed to the largest suitable values in order to achieve the best computational accuracy for the drift compensation process.

Address offset: 0x0110
Reset value: 0x0000 0000

Bit 31  ELT: Enable local time.
0: Local time is stopped, default
1: Local time is enabled

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Note: The local time is started by setting ELT. It remains active until ELT is reset or until the next hardware reset. FDCAN_TURCF.DC is locked when FDCAN_TURCF.ELT = 1. If ELT is written to 0, the readable value will stay at 1 until the new value has been synchronized into the CAN clock domain. During this time write access to the other bits of the register remains locked.

Bit 30  Reserved, must be kept at reset value.

Bits 29:16  DC[13:0]: Denominator configuration.
0x0000: Illegal value
0x0001 to 0x3FFF: Denominator configuration

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:0  NCL[15:0]: Numerator configuration low.
Write access to the TUR numerator configuration low is only possible during configuration with FDCAN_TURCF.ELT = 0 or if FDCAN_TTOCF.EECS (external clock synchronization enabled) is set. When a new value for NCL is written outside TT configuration mode, the new value takes effect when FDCAN_TTOST.WECS is cleared to 0. NCL is locked FDCAN_TTOST.WECS is 1.
0x0000–FFFF Numerator configuration low

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.
Note: If NC < 7 x DC in TTCAN level 1, it is required that subsequent time marks in the trigger memory must differ by at least two NTUs.

### 61.5.52 FDCAN TT operation control register (FDCAN_TTOCN)

Address offset: 0x0114

Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LCKC</td>
<td>Res</td>
<td>ESCN</td>
<td>NIG</td>
<td>TMG</td>
<td>FGP</td>
<td>GCS</td>
<td>TTIE</td>
<td>TMC[1:0]</td>
<td>RTIE</td>
<td>SWS[1:0]</td>
<td>SWP</td>
<td>ECS</td>
<td>SGT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

- **Bit 15 LCKC**: TT operation control register Locked.
  - Set by a write access to register FDCAN_TTOCN. Reset when the updated configuration has been synchronized into the CAN clock domain.
  - 0: Write access to FDCAN_TTOCN enabled
  - 1: Write access to FDCAN_TTOCN locked

- **Bit 14 Reserved, must be kept at reset value.**

- **Bit 13 ESCN**: External synchronization control
  - If enabled the FDCAN synchronizes its cycle time phase to an external event signaled by a rising edge at event trigger pin (see Section 61.4.17: Synchronization to external time schedule).
  - 0: External synchronization disabled
  - 1: External synchronization enabled

- **Bit 12 NIG**: Next is Gap.
  - This bit can only be set when the FDCAN is the actual time master and when it is configured for external event-synchronized time-triggered operation (FDCAN_TTOCF.GEN = 1)
  - 0: No action, reset by reception of any reference message
  - 1: Transmit next reference message with Next_is_Gap = 1

- **Bit 11 TMG**: Time mark Gap.
  - 0: Reset by each reference message
  - 1: Next reference message started when register time mark interrupt FDCAN_TTIR.RTMI is activated

- **Bit 10 FGP**: Finish Gap.
  - Set by the CPU, reset by each reference message
  - 0: No reference message requested
  - 1: Application requested start of reference message
Bit 9 **GCS**: Gap control select
   0: Gap control independent from event trigger
   1: Gap control by input event trigger pin

Bit 8 **TTIE**: Trigger time mark interrupt pulse enable
   External time mark events are configured by trigger memory element TMEX. A trigger time mark interrupt pulse is generated when the trigger memory element becomes active, and the FDCAN is in synchronization state In_Schedule or In_Gap.
   0: Trigger time mark interrupt output fdcan1_tmp for more than one instance and fdcan_tmp if only one instance disabled
   1: Trigger time mark interrupt output fdcan1_tmp for more than one instance and fdcan_tmp if only one instance enabled

Bits 7:6 **TMC[1:0]**: Register time mark Compare.
   00: No Register time mark interrupt generated
   01: Register time mark interrupt if time mark = cycle time
   10: Register time mark interrupt if time mark = local time
   11: Register time mark interrupt if time mark = global time
   Note: When changing the time mark reference (cycle, local, global time), it is recommended to first write TMC = 00, then reconfigure FDCAN_TTTMK, and finally set FDCAN_TMC to the intended time reference.

Bit 5 **RTIE**: Register time mark interrupt pulse enable.
   Register time mark interrupts are configured by register FDCAN_TTTMK. A register time mark interrupt pulse with the length of one fdcan_tq_ck period is generated when time referenced by FDCAN_TTOCN.TMC (cycle, local, or global) equals FDCAN_TTTMK.TM, independent of the synchronization state.
   0: Register time mark interrupt output disabled
   1: Register time mark interrupt output enabled

Bits 4:3 **SWS[1:0]**: Stop watch source.
   00: Stop watch disabled
   01: Actual value of cycle time is copied to FDCAN_TTCPT.SWV
   10: Actual value of local time is copied to FDCAN_TTCPT.SWV
   11: Actual value of global time is copied to FDCAN_TTCPT.SWV

Bit 2 **SWP**: Stop watch polarity.
   0: Rising edge trigger
   1: Falling edge trigger

Bit 1 **ECS**: External clock synchronization.
   Writing a 1 to ECS sets FDCAN_TTOST.WECS if the node is the actual time master. ECS is reset after one APB clock period. The external clock synchronization takes effect at the start of the next basic cycle.

Bit 0 **SGT**: Set global time.
   Writing a 1 to SGT sets FDCAN_TTOST.WGDT if the node is the actual time master. SGT is reset after one APB clock period. The global time preset takes effect when the node transmits the next reference message with the Master_Ref_Mark modified by the preset value written to FDCAN_TTGTP.
61.5.53  **FDCAN TT global time preset register (FDCAN_TTGTP)**

If TTOST.WGDT is set, the next reference message will be transmitted with the Master_Ref_Mark modified by the preset value and with Disc_Bit = 1, presetting the global time in all nodes simultaneously.

TP is reset to 0x0000 each time a reference message with Disc_Bit = 1 becomes valid or if the node is not the current time master. TP is locked while FDCAN_TTOST.WGTD = 1 after setting FDCAN_TTOCN.SGT until the reference message with Disc_Bit = 1 becomes valid or until the node is no longer the current time master.

Address offset: 0x0118

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address offset: 0x0118</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value: 0x0000 0000</td>
</tr>
</tbody>
</table>

**Clinical response:**

If TTOST.WGDT is set, the next reference message will be transmitted with the Master_Ref_Mark modified by the preset value and with Disc_Bit = 1, presetting the global time in all nodes simultaneously.

TP is reset to 0x0000 each time a reference message with Disc_Bit = 1 becomes valid or if the node is not the current time master. TP is locked while FDCAN_TTOST.WGTD = 1 after setting FDCAN_TTOCN.SGT until the reference message with Disc_Bit = 1 becomes valid or until the node is no longer the current time master.

<table>
<thead>
<tr>
<th>Address offset: 0x0118</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value: 0x0000 0000</td>
</tr>
</tbody>
</table>

61.5.54  **FDCAN TT time mark register (FDCAN_TTTMK)**

A time mark interrupt (FDCAN_TTIR.TMI = 1) is generated when the time base indicated by FDCAN_TTOCN.TMC (cycle time, local time, or global time) has the same value as TM.

Address offset: 0x011C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address offset: 0x011C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value: 0x0000 0000</td>
</tr>
</tbody>
</table>
61.5.55 FDCAN TT interrupt register (FDCAN_TTIR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register.

Address offset: 0x0120
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IWTG</td>
<td>ELC</td>
<td>SE2</td>
<td>SE1</td>
<td>TXO</td>
<td>TXU</td>
<td>GTE</td>
<td>GTD</td>
<td>GTW</td>
<td>SWE</td>
<td>TTMi</td>
<td>RTMi</td>
<td>SOG</td>
<td>CSM</td>
<td>SMC</td>
<td>SBC</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 CER: Configuration error.
Trigger out of order.
0: No error found in trigger list
1: Error found in trigger list

Note: When using byte access to register FDCAN_TTTMK it is recommended to first disable the time mark compare function (FDCAN_TTOCN.TMC = 00) to avoid comparisons on inconsistent register values.
Bit 17  **AW**: Application watchdog.
0: Application watchdog served in time
1: Application watchdog not served in time

Bit 16  **WT**: Watch trigger.
0: No missing reference message
1: Missing reference message (level 0: cycle time 0xFF00)

Bit 15  **IWTG**: Initialization watch trigger.
The initialization is restarted by resetting IWT.
0: No missing reference message during system startup
1: No system startup due to missing reference message

Bit 14  **ELC**: Error level Changed.
Not set when error level changed during initialization.
0: No change in error level
1: Error level changed

Bit 13  **SE2**: Scheduling error 2.
0: No scheduling error 2
1: Scheduling error 2 occurred

Bit 12  **SE1**: Scheduling error 1.
0: No scheduling error 1
1: Scheduling error 1 occurred

Bit 11  **TXO**: Tx count overflow.
0: Number of Tx trigger as expected
1: More Tx trigger than expected in one cycle

Bit 10  **TXU**: Tx count underflow.
0: Number of Tx trigger as expected
1: Less Tx trigger than expected in one cycle

Bit 9   **GTE**: Global time error.
Synchronization deviation SD exceeds limit specified by FDCAN_TTOCF.LDSDL, TTCAN level 0, 2 only.
0: Synchronization deviation within limit
1: Synchronization deviation exceeded limit

Bit 8   **GTD**: Global time discontinuity.
0: No discontinuity of global time
1: Discontinuity of global time

Bit 7   **GTW**: Global time wrap
0: No global time wrap occurred
1: Global time wrap from 0xFFFF to 0x0000 occurred

Bit 6   **SWE**: Stop watch event
0: No rising/falling edge at stop watch trigger pin detected
1: Rising/falling edge at stop watch trigger pin detected
Bit 5  **TTMI**: Trigger time mark event internal  
Internal time mark events are configured by trigger memory element TMIN (see Section 61.4.23: FDCAN trigger memory element). Set when the trigger memory element becomes active, and the FDCAN is in synchronization state In_Gap or In_Schedule.  
0: Time mark not reached  
1: Time mark reached (level 0: cycle time FDCAN_TTOCF.RTO x 0x200)

Bit 4  **RTMI**: Register time mark interrupt.  
Set when time referenced by TTOCN.TMC (cycle, local, or global) equals FDCAN_TTTMK.TM, independently from the synchronization state.  
0: Time mark not reached  
1: Time mark reached

Bit 3  **SOG**: Start of Gap  
0: No reference message seen with Next_is_Gap bit set  
1: Reference message with Next_is_Gap bit set becomes valid

Bit 2  **CSM**: Change of synchronization mode.  
0: No change in master to slave relation or schedule synchronization  
1: Master to slave relation or schedule synchronization changed, also set when FDCAN_TTOST.SPL is reset

Bit 1  **SMC**: Start of matrix cycle.  
0: No matrix cycle started since bit has been reset  
1: Matrix cycle started

Bit 0  **SBC**: Start of basic cycle.  
0: No basic cycle started since bit has been reset  
1: Basic cycle started

### 61.5.56 FDCAN TT interrupt enable register (FDCAN_TTIE)

The settings in the TT interrupt enable register determine which status changes in the TT interrupt register will result in an interrupt.

Address offset: 0x0124  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>IWTE</td>
<td>ELCE</td>
<td>SE2E</td>
<td>SE1E</td>
<td>TXOE</td>
<td>TXUE</td>
<td>GTEE</td>
<td>GTDE</td>
<td>GTWE</td>
<td>SWEE</td>
<td>TTME</td>
<td>RTME</td>
<td>SOGE</td>
<td>CSME</td>
<td>SMCE</td>
<td>SBCE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bits 31:19**: Reserved, must be kept at reset value.

- **Bit 18 CERE**: Configuration error interrupt enable  
  0: TT interrupt disabled  
  1: TT interrupt enabled
Bit 17 **AWE**: Application watchdog interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 16 **WTE**: Watch trigger interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 15 **IWTE**: Initialization watch trigger interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 14 **ELCE**: Change error level interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 13 **SE2E**: Scheduling error 2 interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 12 **SE1E**: Scheduling error 1 interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 11 **TXOE**: Tx count overflow interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 10 **TXUE**: Tx count underflow interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 9 **GTEE**: Global time error interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 8 **GTDE**: Global time discontinuity interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 7 **GTWE**: Global time wrap interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 6 **SWEE**: Stop watch event interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 5 **TTMIE**: Trigger time mark event internal interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 4 **RTMIE**: Register time mark interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled
Bit 3 **SOGE**: Start of gap interrupt enable
- 0: TT interrupt disabled
- 1: TT interrupt enabled

Bit 2 **CSME**: Change of synchronization mode interrupt enable
- 0: TT interrupt disabled
- 1: TT interrupt enabled

Bit 1 **SMCE**: Start of matrix cycle interrupt enable
- 0: TT interrupt disabled
- 1: TT interrupt enabled

Bit 0 **SBCE**: Start of basic cycle interrupt enable
- 0: TT interrupt disabled
- 1: TT interrupt enabled

### 61.5.57 FDCAN TT interrupt line select register (FDCAN_TTILS)

The TT interrupt Line select register assigns an interrupt generated by a specific interrupt flag from the TT interrupt register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via FDCAN_ILE.EINT0 and FDCAN_ILE.EINT1.

Address offset: 0x0128
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>CERL: Configuration error interrupt Line</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>AWL: Application watchdog interrupt Line</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>WTL: Watch trigger interrupt Line</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>IWTL: Initialization watch trigger interrupt Line</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>IWTL</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>SE2L</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>SE1L</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>TXUL</td>
<td>nw</td>
</tr>
<tr>
<td>23</td>
<td>TXOL</td>
<td>nw</td>
</tr>
<tr>
<td>22</td>
<td>GTDL</td>
<td>nw</td>
</tr>
<tr>
<td>21</td>
<td>GTWL</td>
<td>nw</td>
</tr>
<tr>
<td>20</td>
<td>SWEL</td>
<td>nw</td>
</tr>
<tr>
<td>19</td>
<td>TTMIL</td>
<td>nw</td>
</tr>
<tr>
<td>18</td>
<td>RTMIL</td>
<td>nw</td>
</tr>
<tr>
<td>17</td>
<td>SOGL</td>
<td>nw</td>
</tr>
<tr>
<td>16</td>
<td>CSML</td>
<td>nw</td>
</tr>
<tr>
<td>15</td>
<td>SMCL</td>
<td>nw</td>
</tr>
<tr>
<td>14</td>
<td>AWL</td>
<td>nw</td>
</tr>
<tr>
<td>13</td>
<td>WTL</td>
<td>nw</td>
</tr>
<tr>
<td>12</td>
<td>IWTL</td>
<td>nw</td>
</tr>
<tr>
<td>11</td>
<td>SE2L</td>
<td>nw</td>
</tr>
<tr>
<td>10</td>
<td>SE1L</td>
<td>nw</td>
</tr>
<tr>
<td>9</td>
<td>TXOL</td>
<td>nw</td>
</tr>
<tr>
<td>8</td>
<td>TXUL</td>
<td>nw</td>
</tr>
<tr>
<td>7</td>
<td>GTDL</td>
<td>nw</td>
</tr>
<tr>
<td>6</td>
<td>GTWL</td>
<td>nw</td>
</tr>
<tr>
<td>5</td>
<td>SWEL</td>
<td>nw</td>
</tr>
<tr>
<td>4</td>
<td>TTMIL</td>
<td>nw</td>
</tr>
<tr>
<td>3</td>
<td>RTMIL</td>
<td>nw</td>
</tr>
<tr>
<td>2</td>
<td>SOGL</td>
<td>nw</td>
</tr>
<tr>
<td>1</td>
<td>CSML</td>
<td>nw</td>
</tr>
<tr>
<td>0</td>
<td>SMCL</td>
<td>nw</td>
</tr>
</tbody>
</table>

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CERL**: Configuration error interrupt Line
- 0: TT interrupt assigned to interrupt line 0
- 1: TT interrupt assigned to interrupt line 1

Bit 17 **AWL**: Application watchdog interrupt Line
- 0: TT interrupt assigned to interrupt line 0
- 1: TT interrupt assigned to interrupt line 1

Bit 16 **WTL**: Watch trigger interrupt Line
- 0: TT interrupt assigned to interrupt line 0
- 1: TT interrupt assigned to interrupt line 1

Bit 15 **IWTL**: Initialization watch trigger interrupt Line
- 0: TT interrupt assigned to interrupt line 0
- 1: TT interrupt assigned to interrupt line 1
Bit 14 **ELCL**: Change error level interrupt Line  
0: TT interrupt assigned to interrupt line 0  
1: TT interrupt assigned to interrupt line 1

Bit 13 **SE2L**: Scheduling error 2 interrupt Line  
0: TT interrupt assigned to interrupt line 0  
1: TT interrupt assigned to interrupt line 1

Bit 12 **SE1L**: Scheduling error 1 interrupt Line  
0: TT interrupt assigned to interrupt line 0  
1: TT interrupt assigned to interrupt line 1

Bit 11 **TXOL**: Tx count overflow interrupt Line  
0: TT interrupt assigned to interrupt line 0  
1: TT interrupt assigned to interrupt line 1

Bit 10 **TXUL**: Tx count underflow interrupt Line  
0: TT interrupt assigned to interrupt line 0  
1: TT interrupt assigned to interrupt line 1

Bit 9 **GTEL**: Global time error interrupt Line  
0: TT interrupt assigned to interrupt line 0  
1: TT interrupt assigned to interrupt line 1

Bit 8 **GTDL**: Global time discontinuity interrupt Line  
0: TT interrupt assigned to interrupt line 0  
1: TT interrupt assigned to interrupt line 1

Bit 7 **GTWL**: Global time wrap interrupt Line  
0: TT interrupt assigned to interrupt line 0  
1: TT interrupt assigned to interrupt line 1

Bit 6 **SWEL**: Stop watch event interrupt Line  
0: TT interrupt assigned to interrupt line 0  
1: TT interrupt assigned to interrupt line 1

Bit 5 **TTMIL**: Trigger time mark event internal interrupt Line  
0: TT interrupt assigned to interrupt line 0  
1: TT interrupt assigned to interrupt line 1

Bit 4 **RTMIL**: Register time mark interrupt Line  
0: TT interrupt assigned to interrupt line 0  
1: TT interrupt assigned to interrupt line 1

Bit 3 **SOGL**: Start of gap interrupt Line  
0: TT interrupt assigned to interrupt line 0  
1: TT interrupt assigned to interrupt line 1

Bit 2 **CSML**: Change of synchronization mode interrupt Line  
0: TT interrupt assigned to interrupt line 0  
1: TT interrupt assigned to interrupt line 1
Bit 1 **SMCL**: Start of matrix cycle interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 0 **SBCL**: Start of basic cycle interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

### 61.5.58 FDCAN TT operation status register (FDCAN_TTOST)

Address offset: 0x012C
Reset value: 0x0000 0080

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td><strong>SPL</strong>: Schedule phase lock</td>
<td>r</td>
<td>0: Phase outside range</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Phase inside range</td>
</tr>
<tr>
<td>30</td>
<td><strong>WECS</strong>: Wait for external clock synchronization</td>
<td>r</td>
<td>0: No external clock synchronization pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Node waits for external clock synchronization to take effect. The bit is reset at the start of the next basic cycle.</td>
</tr>
<tr>
<td>29</td>
<td><strong>AWE</strong>: Application watchdog event</td>
<td>r</td>
<td>0: Application watchdog served in time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Failed to serve application watchdog in time</td>
</tr>
<tr>
<td>28</td>
<td><strong>WFE</strong>: Wait for event</td>
<td>r</td>
<td>0: No Gap announced, reset by a reference message with Next_is_Gap = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Reference message with Next_is_Gap = 1 received</td>
</tr>
<tr>
<td>27</td>
<td><strong>GSI</strong>: Gap started Indicator</td>
<td>r</td>
<td>0: No Gap in schedule, reset by each reference message and for all time slaves</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Gap time after basic cycle has started</td>
</tr>
<tr>
<td>26:24</td>
<td><strong>TMP[2:0]</strong>: Time master priority</td>
<td>r</td>
<td>0x0-7 Priority of actual time master</td>
</tr>
</tbody>
</table>
Bit 23 **GFI**: Gap finished Indicator.
Set when the CPU writes FDCAN_TTOCN.FGP, or by a time mark interrupt if TMG = 1, or via input pin (event trigger) if FDCAN_TTOCN.GCS = 1. Not set by Ref_Trigger_Gap or when Gap is finished by another node sending a reference message.
0: Reset at the end of each reference message
1: Gap finished by FDCAN

Bit 22 **WGTD**: Wait for global time discontinuity.
0: No global time preset pending
1: Node waits for the global time preset to take effect. The bit is reset when the node has transmitted a reference message with Disc_Bit = 1 or after it received a reference message.

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:8 **RTO[7:0]**: Reference trigger offset.
The reference trigger offset value is a signed integer with a range from -127 (0x81) to 127 (0x7F). There is no notification when the lower limit of -127 is reached. In case the FDCAN becomes time master (MS[1:0] = 11), the reset of RTO is delayed due to synchronization between Host and CAN clock domain. For time slaves the value configured by FDCAN_TTOCF.IRTO is read.
0x00-FF Actual reference trigger offset value

Bit 7 **QCS**: Quality of clock Speed.
Only relevant in TTCAN level 0 and level 2, otherwise fixed to 1.
0: Local clock speed not synchronized to time master clock speed
1: Synchronization deviation ≤ SDL

Bit 6 **QGTP**: Quality of global time phase.
Only relevant in TTCAN level 0 and level 2, otherwise fixed to 0.
0: Global time not valid
1: Global time in phase with time master

Bits 5:4 **SYS[1:0]**: Synchronization state
00: Out of Synchronization
01: Synchronizing to FDCAN communication
10: Schedule suspended by Gap (In_Gap)
11: Synchronized to schedule (In_Schedule)

Bits 3:2 **MS[1:0]**: Master state.
00: Master_Off, no master properties relevant
01: Operating as time Slave
10: Operating as backup time master
11: Operating as current time master

Bits 1:0 **EL[1:0]**: Error level.
00: Severity 0 - No error
01: Severity 1 - Warning
10: Severity 2 - error
11: Severity 3 - Severe error

### 61.5.59 FDCAN TUR numerator actual register (FDCAN_TURNNA)

There is no drift compensation in TTCAN level 1 (NAV = NC). In TTCAN level 0 and level 2, the drift between the node local clock and the time master local clock is calculated. The drift
is compensated when the synchronization deviation (difference between NC and the calculated NAV) is lower than 2 * (FDCAN_TTOCF.LDSDL + 5). With FDCAN_TTOCF.LDSDL < 7, this results in a maximum range for NAV of (NC - 0x1000) ≤ NAV ≤ (NC + 0x1000).

Address offset: 0x0130
Reset value: 0x0000 0000

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

Bits 31:18  Reserved, must be kept at reset value.
Bits 17:0  NAV[17:0]: Numerator actual value.
0x0EFFF Illegal value
0x0F000–20FFF Actual numerator value
0x21000 Illegal value

**61.5.60  FDCAN TT local and global time register (FDCAN_TTLGT)**

Address offset: 0x0134
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LT[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:16  GT[15:0]: Global time.
Non-fractional part of the sum of the node local time and its local offset (see Section 61.4.12: Local time, cycle time, global time, and external clock synchronization).
0x0000–FFFF Global time value of FDCAN network

Bits 15:0  LT[15:0]: Local time.
Non-fractional part of local time, incremented once each local NTU (see Section 61.4.12: Local time, cycle time, global time, and external clock synchronization).
0x0000–FFFF Local time value of FDCAN node

**61.5.61  FDCAN TT cycle time and count register (FDCAN_TTCTC)**

Address offset: 0x0138
Reset value: 0x003F 0000

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

CT[15:0]

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

Bits 21:16 **CC[5:0]**: Cycle count.
0x00–3F Number of actual basic cycle in the system matrix

Bits 15:0 **CT[15:0]**: Cycle time
Non-fractional part of the difference of the node local time and Ref_Mark (see Section 61.4.12: Local time, cycle time, global time, and external clock synchronization).
0x0000–FFFF Cycle time value of FDCAN basic cycle

### 61.5.62 FDCAN TT capture time register (FDCAN_TTCPT)

Address offset: 0x013C

Reset value: 0x0000 0000

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

Bits 31:16 **SWV[15:0]**: Stop watch value.
On a rising / falling edge (as configured via FDCAN_TTOCN.SWP) at the stop watch trigger pin, when FDCAN_TTOCN.SWS is different from 00 and FDCAN_TTIR.SWE is 0, the actual time value as selected by FDCAN_TTOCN.SWS (cycle, local, global) is copied to SWV and TFDCAN_TIR.SWE will be set to 1. Capturing of the next stop watch value is enabled by resetting FDCAN_TTIR.SWE.
0x0000–FFFF Captured stop watch value

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

Bits 5:0 **CCV[5:0]**: Cycle count value
Cycle count value captured together with SWV.
0x00–3F Captured cycle count value

### 61.5.63 FDCAN TT cycle sync mark register (FDCAN_TTCM)

Address offset: 0x0140
Reset value: 0x0000 0000

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

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

**CSM[15:0]**

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CSM[15:0]**: Cycle sync mark.
The cycle sync mark is measured in cycle time. It is updated when the reference message becomes valid and retains its value until the next reference message becomes valid.

0x0000–FFFF Captured cycle time

61.5.64 **FDCAN TT trigger select register (FDCAN_TTTS)**
The settings in the FDCAN_TTTS register select the input to be used as event trigger and stop watch trigger.

Address offset: 0x0300

Reset value: 0x0000 0000

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

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

**EVTSEL[1:0]** Event trigger input selection
These bits are used to select the input to be used as event trigger
00: fdcan1_evt0
01: fdcan1_evt1
10: fdcan1_evt2
11: fdcan1_evt3

Bits 3:2 Reserved, must be kept at reset value.

**SWTDEL[1:0]** Stop watch trigger input selection
These bits are used to select the input to be used as stop watch trigger
00: fdcan1.swt0
01: fdcan1.swt1
10: fdcan1.swt2
11: fdcan1.swt3
### 61.5.65 FDCAN register map and reset value table

**Table 487. FDCAN register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>FDCAN_CREL</td>
<td>0x0004</td>
<td>FDCAN_ENDN</td>
</tr>
<tr>
<td>0x0008</td>
<td>Reserved</td>
<td>0x000C</td>
<td>FDCAN_DBTP</td>
</tr>
<tr>
<td>0x0010</td>
<td>FDCAN_TEST</td>
<td>0x0014</td>
<td>FDCAN_RWD</td>
</tr>
<tr>
<td>0x0018</td>
<td>FDCAN_CCCR</td>
<td>0x001C</td>
<td>FDCAN_NBTP</td>
</tr>
<tr>
<td>0x0020</td>
<td>FDCAN_TSCC</td>
<td>0x0024</td>
<td>FDCAN_TSCV</td>
</tr>
<tr>
<td>0x0028</td>
<td>FDCAN_TOCC</td>
<td>0x002C</td>
<td>FDCAN_TOCV</td>
</tr>
<tr>
<td>0x0030</td>
<td>Reserved</td>
<td>0x0040</td>
<td>FDCAN_ECR</td>
</tr>
</tbody>
</table>

#### Reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>FDCAN_CREL</td>
<td>0x0004</td>
<td>FDCAN_ENDN</td>
</tr>
<tr>
<td>0x0008</td>
<td>Reserved</td>
<td>0x000C</td>
<td>FDCAN_DBTP</td>
</tr>
<tr>
<td>0x0010</td>
<td>FDCAN_TEST</td>
<td>0x0014</td>
<td>FDCAN_RWD</td>
</tr>
<tr>
<td>0x0018</td>
<td>FDCAN_CCCR</td>
<td>0x001C</td>
<td>FDCAN_NBTP</td>
</tr>
<tr>
<td>0x0020</td>
<td>FDCAN_TSCC</td>
<td>0x0024</td>
<td>FDCAN_TSCV</td>
</tr>
<tr>
<td>0x0028</td>
<td>FDCAN_TOCC</td>
<td>0x002C</td>
<td>FDCAN_TOCV</td>
</tr>
<tr>
<td>0x0030</td>
<td>Reserved</td>
<td>0x0040</td>
<td>FDCAN_ECR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>FDCAN_CREL</td>
<td>0x0004</td>
<td>FDCAN_ENDN</td>
</tr>
<tr>
<td>0x0008</td>
<td>Reserved</td>
<td>0x000C</td>
<td>FDCAN_DBTP</td>
</tr>
<tr>
<td>0x0010</td>
<td>FDCAN_TEST</td>
<td>0x0014</td>
<td>FDCAN_RWD</td>
</tr>
<tr>
<td>0x0018</td>
<td>FDCAN_CCCR</td>
<td>0x001C</td>
<td>FDCAN_NBTP</td>
</tr>
<tr>
<td>0x0020</td>
<td>FDCAN_TSCC</td>
<td>0x0024</td>
<td>FDCAN_TSCV</td>
</tr>
<tr>
<td>0x0028</td>
<td>FDCAN_TOCC</td>
<td>0x002C</td>
<td>FDCAN_TOCV</td>
</tr>
<tr>
<td>0x0030</td>
<td>Reserved</td>
<td>0x0040</td>
<td>FDCAN_ECR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>FDCAN_CREL</td>
<td>0x0004</td>
<td>FDCAN_ENDN</td>
</tr>
<tr>
<td>0x0008</td>
<td>Reserved</td>
<td>0x000C</td>
<td>FDCAN_DBTP</td>
</tr>
<tr>
<td>0x0010</td>
<td>FDCAN_TEST</td>
<td>0x0014</td>
<td>FDCAN_RWD</td>
</tr>
<tr>
<td>0x0018</td>
<td>FDCAN_CCCR</td>
<td>0x001C</td>
<td>FDCAN_NBTP</td>
</tr>
<tr>
<td>0x0020</td>
<td>FDCAN_TSCC</td>
<td>0x0024</td>
<td>FDCAN_TSCV</td>
</tr>
<tr>
<td>0x0028</td>
<td>FDCAN_TOCC</td>
<td>0x002C</td>
<td>FDCAN_TOCV</td>
</tr>
<tr>
<td>0x0030</td>
<td>Reserved</td>
<td>0x0040</td>
<td>FDCAN_ECR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>FDCAN_CREL</td>
<td>0x0004</td>
<td>FDCAN_ENDN</td>
</tr>
<tr>
<td>0x0008</td>
<td>Reserved</td>
<td>0x000C</td>
<td>FDCAN_DBTP</td>
</tr>
<tr>
<td>0x0010</td>
<td>FDCAN_TEST</td>
<td>0x0014</td>
<td>FDCAN_RWD</td>
</tr>
<tr>
<td>0x0018</td>
<td>FDCAN_CCCR</td>
<td>0x001C</td>
<td>FDCAN_NBTP</td>
</tr>
<tr>
<td>0x0020</td>
<td>FDCAN_TSCC</td>
<td>0x0024</td>
<td>FDCAN_TSCV</td>
</tr>
<tr>
<td>0x0028</td>
<td>FDCAN_TOCC</td>
<td>0x002C</td>
<td>FDCAN_TOCV</td>
</tr>
<tr>
<td>0x0030</td>
<td>Reserved</td>
<td>0x0040</td>
<td>FDCAN_ECR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>FDCAN_CREL</td>
<td>0x0004</td>
<td>FDCAN_ENDN</td>
</tr>
<tr>
<td>0x0008</td>
<td>Reserved</td>
<td>0x000C</td>
<td>FDCAN_DBTP</td>
</tr>
<tr>
<td>0x0010</td>
<td>FDCAN_TEST</td>
<td>0x0014</td>
<td>FDCAN_RWD</td>
</tr>
<tr>
<td>0x0018</td>
<td>FDCAN_CCCR</td>
<td>0x001C</td>
<td>FDCAN_NBTP</td>
</tr>
<tr>
<td>0x0020</td>
<td>FDCAN_TSCC</td>
<td>0x0024</td>
<td>FDCAN_TSCV</td>
</tr>
<tr>
<td>0x0028</td>
<td>FDCAN_TOCC</td>
<td>0x002C</td>
<td>FDCAN_TOCV</td>
</tr>
<tr>
<td>0x0030</td>
<td>Reserved</td>
<td>0x0040</td>
<td>FDCAN_ECR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>FDCAN_CREL</td>
<td>0x0004</td>
<td>FDCAN_ENDN</td>
</tr>
<tr>
<td>0x0008</td>
<td>Reserved</td>
<td>0x000C</td>
<td>FDCAN_DBTP</td>
</tr>
<tr>
<td>0x0010</td>
<td>FDCAN_TEST</td>
<td>0x0014</td>
<td>FDCAN_RWD</td>
</tr>
<tr>
<td>0x0018</td>
<td>FDCAN_CCCR</td>
<td>0x001C</td>
<td>FDCAN_NBTP</td>
</tr>
<tr>
<td>0x0020</td>
<td>FDCAN_TSCC</td>
<td>0x0024</td>
<td>FDCAN_TSCV</td>
</tr>
<tr>
<td>0x0028</td>
<td>FDCAN_TOCC</td>
<td>0x002C</td>
<td>FDCAN_TOCV</td>
</tr>
<tr>
<td>0x0030</td>
<td>Reserved</td>
<td>0x0040</td>
<td>FDCAN_ECR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>FDCAN_CREL</td>
<td>0x0004</td>
<td>FDCAN_ENDN</td>
</tr>
<tr>
<td>0x0008</td>
<td>Reserved</td>
<td>0x000C</td>
<td>FDCAN_DBTP</td>
</tr>
<tr>
<td>0x0010</td>
<td>FDCAN_TEST</td>
<td>0x0014</td>
<td>FDCAN_RWD</td>
</tr>
<tr>
<td>0x0018</td>
<td>FDCAN_CCCR</td>
<td>0x001C</td>
<td>FDCAN_NBTP</td>
</tr>
<tr>
<td>0x0020</td>
<td>FDCAN_TSCC</td>
<td>0x0024</td>
<td>FDCAN_TSCV</td>
</tr>
<tr>
<td>0x0028</td>
<td>FDCAN_TOCC</td>
<td>0x002C</td>
<td>FDCAN_TOCV</td>
</tr>
<tr>
<td>0x0030</td>
<td>Reserved</td>
<td>0x0040</td>
<td>FDCAN_ECR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>FDCAN_CREL</td>
<td>0x0004</td>
<td>FDCAN_ENDN</td>
</tr>
<tr>
<td>0x0008</td>
<td>Reserved</td>
<td>0x000C</td>
<td>FDCAN_DBTP</td>
</tr>
<tr>
<td>0x0010</td>
<td>FDCAN_TEST</td>
<td>0x0014</td>
<td>FDCAN_RWD</td>
</tr>
<tr>
<td>0x0018</td>
<td>FDCAN_CCCR</td>
<td>0x001C</td>
<td>FDCAN_NBTP</td>
</tr>
<tr>
<td>0x0020</td>
<td>FDCAN_TSCC</td>
<td>0x0024</td>
<td>FDCAN_TSCV</td>
</tr>
<tr>
<td>0x0028</td>
<td>FDCAN_TOCC</td>
<td>0x002C</td>
<td>FDCAN_TOCV</td>
</tr>
<tr>
<td>0x0030</td>
<td>Reserved</td>
<td>0x0040</td>
<td>FDCAN_ECR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>FDCAN_CREL</td>
<td>0x0004</td>
<td>FDCAN_ENDN</td>
</tr>
<tr>
<td>0x0008</td>
<td>Reserved</td>
<td>0x000C</td>
<td>FDCAN_DBTP</td>
</tr>
<tr>
<td>0x0010</td>
<td>FDCAN_TEST</td>
<td>0x0014</td>
<td>FDCAN_RWD</td>
</tr>
<tr>
<td>0x0018</td>
<td>FDCAN_CCCR</td>
<td>0x001C</td>
<td>FDCAN_NBTP</td>
</tr>
<tr>
<td>0x0020</td>
<td>FDCAN_TSCC</td>
<td>0x0024</td>
<td>FDCAN_TSCV</td>
</tr>
<tr>
<td>0x0028</td>
<td>FDCAN_TOCC</td>
<td>0x002C</td>
<td>FDCAN_TOCV</td>
</tr>
<tr>
<td>0x0030</td>
<td>Reserved</td>
<td>0x0040</td>
<td>FDCAN_ECR</td>
</tr>
</tbody>
</table>
### Table 487. FDCAN register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0044</td>
<td>FDCAN_PSR</td>
<td>TDCV[6:0]</td>
</tr>
<tr>
<td>0x0048</td>
<td>FDCAN_TDCR</td>
<td>TDCO[6:0]</td>
</tr>
<tr>
<td>0x0050</td>
<td>FDCAN_IR</td>
<td>TDCF[6:0]</td>
</tr>
<tr>
<td>0x0054</td>
<td>FDCAN_IE</td>
<td>FDCAN_TDCR</td>
</tr>
<tr>
<td>0x0058</td>
<td>FDCAN_ILS</td>
<td>FDCAN_GFC</td>
</tr>
<tr>
<td>0x005C</td>
<td>FDCAN_ILE</td>
<td>FDCAN_PSR</td>
</tr>
<tr>
<td>0x0060</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x007F</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0080</td>
<td>FDCAN_GFC</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0084</td>
<td>FDCAN_SIFDC</td>
<td>LSS[7:0]</td>
</tr>
<tr>
<td>0x0088</td>
<td>FDCAN_SIFC</td>
<td>FLSSA[13:0]</td>
</tr>
<tr>
<td>0x0090</td>
<td>FDCAN_XIDAM</td>
<td>LSE[7:0]</td>
</tr>
<tr>
<td>0x0092</td>
<td>FDCAN_XIDC</td>
<td>FLESA[13:0]</td>
</tr>
<tr>
<td>0x0094</td>
<td>FDCAN_XIFC</td>
<td>EIDM[28:0]</td>
</tr>
<tr>
<td>0x0098</td>
<td>FDCAN_XIF1</td>
<td>FDX[8:0]</td>
</tr>
<tr>
<td>0x009C</td>
<td>FDCAN_XIF2</td>
<td>BIDX[5:0]</td>
</tr>
</tbody>
</table>

Reset values:
- FDCAN_PSR: 0x00000000000000000000000000000000
- FDCAN_TDCR: 0x00000000000000000000000000000000
- FDCAN_IR: 0x00000000000000000000000000000000
- FDCAN_IE: 0x00000000000000000000000000000000
- FDCAN_ILS: 0x00000000000000000000000000000000
- FDCAN_ILE: 0x00000000000000000000000000000000
- Reserved: 0x00000000000000000000000000000000
- FDCAN_GFC: 0x00000000000000000000000000000000
- FDCAN_SIFDC: 0x00000000000000000000000000000000
- FDCAN_SIFC: 0x00000000000000000000000000000000
- FDCAN_XIDAM: 0x00000000000000000000000000000000
- FDCAN_XIFC: 0x00000000000000000000000000000000
- FDCAN_XIF1: 0x00000000000000000000000000000000
- FDCAN_XIF2: 0x00000000000000000000000000000000

**Note:** The table continues with additional registers and reset values, but the full table is not provided here. Please refer to the document for complete details.
### Table 487. FDCAN register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>FOM</th>
<th>F0WM[6:0]</th>
<th>F0S[6:0]</th>
<th>F0SA[13:0]</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00A0</td>
<td>FDCAN_RXF0C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00A4</td>
<td>FDCAN_RXF0S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00A8</td>
<td>FDCAN_RXF0SA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00AC</td>
<td>FDCAN_RXBC</td>
<td></td>
<td>RBSA[13:0]</td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00B0</td>
<td>FDCAN_RXF1C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00B4</td>
<td>FDCAN_RXF1S</td>
<td></td>
<td>DMS[1:0]</td>
<td></td>
<td>F1S[6:0]</td>
<td>F1SA[13:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00B8</td>
<td>FDCAN_RXF1A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00BC</td>
<td>FDCAN_RXESC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C0</td>
<td>FDCAN_TXBC</td>
<td></td>
<td>TFQM</td>
<td></td>
<td>NDTB[5:0]</td>
<td>TBSA[13:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C4</td>
<td>FDCAN_TXFQS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C8</td>
<td>FDCAN_TXESC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00CC</td>
<td>FDCAN_TXBRP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00D0</td>
<td>FDCAN_TXBAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The table continues with similar entries for other registers with their respective reset values.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00D4</td>
<td>FDCAN__TXBCHR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00D8</td>
<td>FDCAN__TXBTO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00D0</td>
<td>FDCAN__TXBCF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00E0</td>
<td>FDCAN__TXBIE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00E4</td>
<td>FDCAN__TXBIE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00F0</td>
<td>FDCAN__TXEFC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00F4</td>
<td>FDCAN__TXEF5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00F8</td>
<td>FDCAN__TXEFA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0100</td>
<td>FDCAN__TTTMC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0104</td>
<td>FDCAN__TTRMC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0108</td>
<td>FDCAN__ TTCF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010C</td>
<td>FDCAN__TTTLM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0110</td>
<td>FDCAN__TURCF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0114</td>
<td>FDCAN__TTOCN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reset value: 00000000000000000000000000000000
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset Register name</th>
<th>Offset Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0118</td>
<td>FDCAN_TTGTTP</td>
<td>CTP[15:0]</td>
<td>TP[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x011C FDCAN_TTMK</td>
<td>TICC[6:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TM[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0120 FDCAN_TTIR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0124 FDCAN_TTIE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0128 FDCAN_TTILS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x012C FDCAN_TTOST</td>
<td>RTO[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0130 FDCAN_TURNAA</td>
<td>NAV[17:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0134 FDCAN_TTILT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0138 FDCAN_TTTC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x013C FDCAN_TTCTP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0140 FDCAN_TTCSM</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0144 FDCAN_TTTS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x01FC Reserved</td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 128 for the register boundary addresses.
61.6 **CCU registers**

61.6.1 **Clock calibration unit core release register (FDCAN_CCU_CREL)**

Address offset: 0x0000
Reset value: 0x1114 1218

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

Bits 31:28 **REL[3:0]** Core release = 1
Bits 27:24 **STEP[3:0]** Step of core release = 1
Bits 23:20 **SUBSTEP[3:0]** Sub-step of core release = 1
Bits 19:16 **YEAR[3:0]** Timestamp year =
Bits 15:8 **MON[7:0]** Timestamp month = 12
Bits 7:0 **DAY[7:0]** Timestamp day = 18

61.6.2 **Calibration configuration register (FDCAN_CCU_CCFG)**

Address offset: 0x0004
Reset value: 0x0000 0004

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

Bit 31 **SWR**: Software Reset
Writing a 1 to this bit will reset the calibration FSM to state Not_Calibrated
(FDCAN_CCU_CSTAT.CALS = 00). The calibration watchdog value CWD.WDV is also reset.
Registers FDCAN_CCFG, FDCAN_CCU_CSTAT and the calibration watchdog
configuration CWD.WDC are unchanged. The bit remains set until reset is completed.
Write access by the Host CPU to registers/bits marked with "P=Protected Write" is
possible only when the FDCAN control bits FDCAN_CCCR.CCE = 1 AND
FDCAN_CCCR.INIT = 1.

Bits 30:20 Reserved, must be kept at reset value.
Bits 19:16 **CDIV[3:0]**: Clock Divider

The clock divider has to be configured when the clock calibration is bypassed (BCC = 1) to ensure that the FDCAN requirement is fulfilled.

0000: Divide by 1  
0001: Divide by 2  
0010: Divide by 4  
0011: Divide by 6  
0100: Divide by 8  
0101: Divide by 10  
0110: Divide by 12  
0111: Divide by 14  
1000: Divide by 16  
1001: Divide by 18  
1010: Divide by 20  
1011: Divide by 22  
1100: Divide by 24  
1101: Divide by 26  
1110: Divide by 28  
1111: Divide by 30  

Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when the FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bits 15:8 **OCPM[7:0]**: Oscillator clock periods minimum

Configures the minimum number of periods in two CAN bit times. OCPM is used in basic calibration to avoid false measurements in case of glitches on the bus line. The configured number of periods is OCPM × 32. The configuration depends on the frequency and the bitrate configured in FDCAN modules (from 125 kbit/s up to 1 Mbit/s). It is recommended to configure a value slightly below two CAN bit times. The reset value is 1.6 bit times at 80 MHz fdcan_ker_ck and 1 Mbit/s CAN bitrate.

Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when the FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bit 7 **CFL**: Calibration Field length

0: Calibration field length is 32 bits  
1: Calibration field length is 64 bits  

Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when the FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bit 6 **BCC**: Bypass clock calibration

If this bit is set, the clock input fdcan_ker_ck is routed to the time quanta clock through a clock divider configurable via CDIV, cu_cok is always 1. In this case the baudrate prescaler of the connected FDCANs has to be configured to generate the FDCAN internal time quanta clock.

0: Clock calibration unit generates time quanta clock  
1: Clock calibration unit bypassed (default configuration)
61.6.3 Calibration status register (FDCAN_CCU_CSTAT)

Address offset: 0x0008
Reset value: 0x0203 FFFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALS[1:0]</td>
<td>Res.</td>
<td>TQC[10:0]</td>
<td>OCPC[17:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Bits 31:30 **CALS[1:0]**: Calibration state
00: Not_Calibrated
01: Basic_Calibrated
10: Precision_Calibrated
11: Reserved

Bit 29 Reserved, must be kept at reset value.

Bits 28:18 **TQC[10:0]**: Time quanta counter
Captured number of time quanta in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

Bits 17:0 **OCPC[17:0]**: Oscillator clock period counter
Captured number of oscillator clock periods in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

61.6.4 Calibration watchdog register (FDCAN_CCU_CWD)

Address offset: 0x000C
Reset value: 0x0000 0000

The calibration watchdog is started after the first falling edge when the calibration FSM is in state Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00). In this state the calibration watchdog monitors the message received. In case no message was received until the calibration watchdog has counted down to 0, the calibration FSM stays in state Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00), the counter is reloaded with FDCAN_RWD.WDC and basic calibration is restarted after the next falling edge.
When in state Basic_Calibrated (FDCAN_CCU_CSTAT.CALS = 01), the calibration watchdog is restarted with each received message. In case no message was received until the calibration watchdog has counted down to 0, the calibration FSM returns to state Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00), the counter is reloaded with FDCAN_RWD.WDC and basic calibration is restarted after the next falling edge.

When a quartz message is received, state Precision_Calibrated (FDCAN_CCU_CSTAT.CALS = 10) is entered and the calibration watchdog is restarted. In this state the calibration watchdog monitors the quartz message received input. In case no message from a quartz controlled node is received by the attached TTCAN until the calibration watchdog has counted down to 0, the calibration FSM transits back to state Basic_Calibrated (FDCAN_CCU_CSTAT.CALS = 01). The signal is active when the CAN protocol engine on the attached TTCAN is started i.e. when the INIT bit is reset.

A calibration watchdog event also sets interrupt flag FDCAN_CCU_IR.CWE. If enabled by FDCAN_CCU_IE.CWEE, interrupt line is activated (set to high). Interrupt line remains active until interrupt flag FDCAN_CCU_IR.CWE is reset.

61.6.5 Clock calibration unit interrupt register (FDCAN_CCU_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register. The configuration of FDCAN_CCU_IE controls whether an interrupt is generated or not.

Address offset: 0x0010
Reset value: 0x0000 0000
61.6.6 **Clock calibration unit interrupt enable register (FDCAN_CCU_IE)**

Address offset: 0x0014  
Reset value: 0x0000 0000

The settings in the CU interrupt enable register determine whether a status change in the CU interrupt register will be signaled on an interrupt line.

<table>
<thead>
<tr>
<th>Bits 31:2 Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 1 **CSC**: Calibration state changed  
0: Calibration state unchanged  
1: Calibration state has changed |
| Bit 0 **CWE**: Calibration watchdog event  
0: No calibration watchdog event  
1: Calibration watchdog event occurred |

### Clock calibration unit interrupt enable register (FDCAN_CCU_IE)

<table>
<thead>
<tr>
<th>Bits 31:2 Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 1 **CSCE**: Calibration state changed enable  
0: Interrupt disabled  
1: Interrupt enabled |
| Bit 0 **CWEE**: Calibration watchdog event enable  
0: Interrupt disabled  
1: Interrupt enabled |
### CCU register map and reset value table

Table 488. CCU register map and reset values

| Offset | Register name          | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|-----------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x0000 | FDCAN_CCU_CREL        | REL | 3   | 0   | 1   | 0   | 0   | 0   | 1   | 0   | 0   | 1   | 0   | 0   | 0   | 1   | 0   | 0   | 1   | 0   | 0   | 0   | 1   | 1   | 0   | 0   |     |     |     |     |     |
| Reset value |                  | 0   | 0   | 0   | 1   | 0   | 0   | 0   | 1   | 0   | 0   | 1   | 0   | 0   | 0   | 1   | 0   | 0   | 1   | 0   | 0   | 0   | 1   | 1   | 0   | 0   |     |     |     |     |     |
| 0x0004 | FDCAN_CCU_CCFG       | SWR |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| Reset value |                  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |     |     |     |     |     |
| 0x0008 | FDCAN_CCU_CSTAT      | CSL |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| Reset value |                  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |     |     |     |     |     |
| 0x000C | FDCAN_CCU_CWD        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| Reset value |                  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |     |     |     |     |     |
| 0x0010 | FDCAN_CCU_IR         |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| Reset value |                  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |     |     |     |     |     |
| 0x0014 | FDCAN_CCU_IE         |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| Reset value |                  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |     |     |     |     |     |

Refer to Section 2.3 on page 128 for the register boundary addresses.
62 USB on-the-go high-speed (OTG_HS)

62.1 Introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

This section presents the architecture and the programming model of the OTG_HS controller.

The following acronyms are used throughout the section:

- FS: Full-speed
- LS: Low-speed
- HS: High-speed
- MAC: Media access controller
- OTG: On-the-go
- PFC: Packet FIFO controller
- PHY: Physical layer
- USB: Universal serial bus
- UTMI: USB 2.0 Transceiver Macrocell interface (UTMI)
- ULPI: UTMI+ Low Pin Interface
- LPM: Link power management
- BCD: Battery charging detector
- HNP: Host negotiation protocol
- SRP: Session request protocol

References are made to the following documents:

- USB On-The-Go Supplement, Revision 2.0
- Universal Serial Bus Revision 2.0 Specification
- USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
- Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
- Battery Charging Specification, Revision 1.2

The USB OTG is a dual-role device (DRD) controller that supports both device and host functions and is fully compliant with the On-The-Go Supplement to the USB 2.0 Specification. It can also be configured as a host-only or device-only controller, fully compliant with the USB 2.0 Specification. OTG_HS supports the speeds defined in the Table 489: OTG_HS speeds supported below. The USB OTG supports both HNP and SRP. The only external device required is a charge pump for VBUS in OTG mode.
62.2 OTG_HS main features

The main features can be divided into three categories: general, host-mode and device-mode features.

62.2.1 General features

The OTG_HS interface general features are the following:

- It is USB-IF certified to the Universal Serial Bus Specification Rev 2.0
- OTG_HS supports the following PHY interfaces:
  - An on-chip full-speed PHY
  - An ULPI interface for external high-speed PHY
- It includes full support (PHY) for the optional On-The-Go (OTG) protocol detailed in the On-The-Go Supplement Rev 2.0 specification
  - Integrated support for A-B device identification (ID line)
  - Integrated support for host Negotiation protocol (HNP) and session request protocol (SRP)
  - It allows host to turn VBUS off to conserve battery power in OTG applications
  - It supports OTG monitoring of VBUS levels with internal comparators
  - It supports dynamic host-peripheral switch of role
- It is software-configurable to operate as:
  - SRP capable USB HS Peripheral (B-device)
  - SRP capable USB HS/LS host (A-device)
  - USB On-The-Go Full-Speed Dual Role device
- It supports HS SOF and LS Keep-alives with
  - SOF pulse PAD connectivity
  - SOF pulse internal connection to timer (TIMx)
  - Configurable framing period
  - Configurable end of frame interrupt
- OTG_HS embeds an internal DMA with thresholding support and software selectable AHB burst type in DMA mode.
- It supports Descriptor-Based Scatter/Gather DMA controller for device and host mode. (Descriptor-Based Congruent-Sequential DMA is not supported). Scatter/Gather DMA operation is supported in both device and host mode. This feature will improve performance for device mode isochronous endpoints. Note that hubs (split transfers)
are not supported in host scatter/gather DMA mode of operation. Split transfers are supported only in host buffer DMA (internal DMA) mode of operation.

- It includes power saving features such as system stop during USB suspend, switch-off of clock domains internal to the digital core, PHY and DFIFO power management.
- It features a dedicated RAM of 4 Kbytes with advanced FIFO control:
  - Configurable partitioning of RAM space into different FIFOs for flexible and efficient use of RAM
  - Each FIFO can hold multiple packets
  - Dynamic memory allocation
  - Configurable FIFO sizes that are not powers of 2 to allow the use of contiguous memory locations
- It guarantees max USB bandwidth for up to one frame (1 ms) without system intervention.
- It supports charging port detection as described in Battery Charging Specification Revision 1.2 on the FS PHY transceiver only.

### 62.2.2 Host-mode features

The OTG_HS interface main features and requirements in host-mode are the following:

- External charge pump for $V_{BUS}$ voltage generation.
- Up to 16 host channels (pipes): each channel is dynamically reconfigurable to allocate any type of USB transfer.
- Built-in hardware scheduler holding:
  - Up to 16 interrupt plus isochronous transfer requests in the periodic hardware queue
  - Up to 16 control plus bulk transfer requests in the non-periodic hardware queue
- Management of a shared Rx FIFO, a periodic Tx FIFO and a nonperiodic Tx FIFO for efficient usage of the USB data RAM.

### 62.2.3 Peripheral-mode features

The OTG_HS interface main features in peripheral-mode are the following:

- 1 bidirectional control endpoint0
- 8 IN endpoints (EPs) configurable to support bulk, interrupt or isochronous transfers
- 8 OUT endpoints configurable to support bulk, interrupt or isochronous transfers
- Management of a shared Rx FIFO and a Tx-OUT FIFO for efficient usage of the USB data RAM
- Management of up to 9 dedicated Tx-IN FIFOs (one for each active IN EP) to put less load on the application
- Support for the soft disconnect feature.
62.3 OTG_HS implementation

Table 490. OTG_HS implementation\(^{(1)}\)

<table>
<thead>
<tr>
<th>USB features</th>
<th>OTG_HS1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device bidirectional endpoints (including EP0)</td>
<td>9</td>
</tr>
<tr>
<td>Host mode channels</td>
<td>16</td>
</tr>
<tr>
<td>Size of dedicated SRAM</td>
<td>4 KB</td>
</tr>
<tr>
<td>USB 2.0 link power management (LPM) support</td>
<td>X</td>
</tr>
<tr>
<td>OTG revision supported</td>
<td>2.0</td>
</tr>
<tr>
<td>Attach detection protocol (ADP) support</td>
<td>-</td>
</tr>
<tr>
<td>Battery charging detection (BCD) support</td>
<td>X</td>
</tr>
<tr>
<td>ULPI available to primary IOs via, muxing</td>
<td>X</td>
</tr>
<tr>
<td>Integrated PHY</td>
<td>FS</td>
</tr>
<tr>
<td>Scatter/gather DMA</td>
<td>X</td>
</tr>
</tbody>
</table>

1. "X" = supported, "-" = not supported.

62.4 OTG_HS functional description

62.4.1 OTG_HS block diagram

Figure 715. OTG_HS high-speed block diagram

MSv40734V2
62.4.2 OTG_HS pin and internal signals

Table 491. OTG_HS input/output pins

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_HS_DP</td>
<td>Digital input/output</td>
<td>USB OTG D+ line</td>
</tr>
<tr>
<td>OTG_HS_DM</td>
<td>Digital input/output</td>
<td>USB OTG D- line</td>
</tr>
<tr>
<td>OTG_HS_ID</td>
<td>Digital input</td>
<td>USB OTG ID</td>
</tr>
<tr>
<td>OTG_HS_VBUS</td>
<td>Analog input</td>
<td>USB OTG VBUS</td>
</tr>
<tr>
<td>OTG_HS_SOF</td>
<td>Digital output</td>
<td>USB OTG Start Of Frame (visibility)</td>
</tr>
<tr>
<td>OTG_HS_ULPI_CK</td>
<td>Digital input</td>
<td>USB OTG ULPI clock</td>
</tr>
<tr>
<td>OTG_HS_ULPI_DIR</td>
<td>Digital input</td>
<td>USB OTG ULPI data bus direction control</td>
</tr>
<tr>
<td>OTG_HS_ULPI_STP</td>
<td>Digital output</td>
<td>USB OTG ULPI data stream stop</td>
</tr>
<tr>
<td>OTG_HS_ULPI_NXT</td>
<td>Digital input</td>
<td>USB OTG ULPI next data stream request</td>
</tr>
<tr>
<td>OTG_HS_ULPI_D[0..7]</td>
<td>Digital input/output</td>
<td>USB OTG ULPI 8-bit bi-directional data bus</td>
</tr>
</tbody>
</table>

Table 492. OTG_HS input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usb_sof</td>
<td>Digital output</td>
<td>USB OTG start-of-frame event for on chip peripherals</td>
</tr>
<tr>
<td>usb_wkup</td>
<td>Digital output</td>
<td>USB OTG wakeup event output</td>
</tr>
<tr>
<td>usb_gbl_it</td>
<td>Digital output</td>
<td>USB OTG global interrupt</td>
</tr>
<tr>
<td>usb_ep1_in_it</td>
<td>Digital output</td>
<td>USB OTG endpoint 1 in interrupt</td>
</tr>
<tr>
<td>usb_ep1_out_it</td>
<td>Digital output</td>
<td>USB OTG endpoint 1 out interrupt</td>
</tr>
</tbody>
</table>

62.4.3 OTG_HS core

The USB OTG receives the 48 MHz clock from the reset and clock controller (RCC). The USB clock is used for driving the 48 MHz domain at full-speed (12 Mbit/s) and must be enabled prior to configuring the OTG core.

The CPU reads and writes from/to the OTG core registers through the AHB peripheral bus. It is informed of USB events through the single USB OTG interrupt line described in Section 62.12: OTG_HS interrupts.

The CPU submits data over the USB by writing 32-bit words to dedicated OTG locations (push registers). The data are then automatically stored into Tx-data FIFOs configured within the USB data RAM. There is one Tx FIFO push register for each in-endpoint (peripheral mode) or out-channel (host mode).

The CPU receives the data from the USB by reading 32-bit words from dedicated OTG addresses (pop registers). The data are then automatically retrieved from a shared Rx FIFO configured within the 4-Kbyte USB data RAM. There is one Rx FIFO pop register for each out-endpoint or in-channel.

The USB protocol layer is driven by the serial interface engine (SIE) and serialized over the USB by the transceiver module within the on-chip physical layer (PHY) or external HS PHY.
62.4.4 Embedded full speed OTG PHY

The full-speed OTG PHY includes the following components:

- FS/LS transceiver module used by both host and device. It directly drives transmission and reception on the single-ended USB lines.
- Integrated ID pull-up resistor used to sample the ID line for A/B device identification.
- DP/DM integrated pull-up and pull-down resistors controlled by the OTG_HS core depending on the current role of the device. As a peripheral, it enables the DP pull-up resistor to signal full-speed peripheral connections as soon as \( V_{BUS} \) is sensed to be at a valid level (B-session valid). In host mode, pull-down resistors are enabled on both DP/DM. Pull-up and pull-down resistors are dynamically switched when the peripheral role is changed via the host negotiation protocol (HNP).
- Pull-up/pull-down resistor ECN circuit. The DP pull-up consists of 2 resistors controlled separately from the OTG_HS as per the resistor Engineering Change Notice applied to USB Rev2.0. The dynamic trimming of the DP pull-up strength allows to achieve a better noise rejection and Tx/Rx signal quality.
- \( V_{BUS} \) sensing comparators with hysteresis used to detect \( V_{BUS} \) valid, A-B session valid and session-end voltage thresholds. They are used to drive the session request protocol (SRP), detect valid startup and end-of-session conditions, and constantly monitor the \( V_{BUS} \) supply during USB operations.

To guarantee a correct operation for the USB OTG_HS peripheral, the AHB frequency should be higher than 30 MHz.

62.4.5 High-speed OTG PHY

The USB OTG_HS core includes an ULPI interface to connect an external HS PHY.

62.5 OTG_HS dual role device (DRD)

62.5.1 ID line detection

The host or peripheral (the default) role is assumed depending on the ID input pin. The ID line status is determined on plugging in the USB cable, depending on whether a MicroA or MicroB plug is connected to the micro-AB receptacle.

- If the B-side of the USB cable is connected with a floating ID wire, the integrated pull-up resistor detects a high ID level and the default peripheral role is confirmed. In this configuration the OTG_HS complies with the standard FSM described in section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.
- If the A-side of the USB cable is connected with a grounded ID, the OTG_HS issues an ID line status change interrupt (CIDSCHG bit in OTG_GINTSTS) for host software initialization, and automatically switches to the host role. In this configuration the OTG_HS complies with the standard FSM described by section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.

62.5.2 HNP dual role device

The HNP capable bit in the Global USB configuration register (HNPCAP bit in OTG_GUSBCFG) enables the OTG_HS core to dynamically change its role from A-host to A-peripheral and vice-versa, or from B-Peripheral to B-host and vice-versa according to the
host negotiation protocol (HNP). The current device status can be read by the combined values of the connector ID status bit in the Global OTG control and status register (CIDSTS bit in OTG_GOTGCTL) and the current mode of operation bit in the global interrupt and status register (CMOD bit in OTG_GINTSTS).

The HNP program model is described in detail in Section 62.15: OTG_HS programming model.

62.5.3 SRP dual role device

The SRP capable bit in the global USB configuration register (SRPCAP bit in OTG_GUSBCFG) enables the OTG_HS core to switch off the generation of VBUS for the A-device to save power. Note that the A-device is always in charge of driving VBUS regardless of the host or peripheral role of the OTG_HS.

The SRP A/B-device program model is described in detail in Section 62.15: OTG_HS programming model.

62.6 OTG_HS as a USB peripheral

This section gives the functional description of the OTG_HS in the USB peripheral mode. The OTG_HS works as an USB peripheral in the following circumstances:

- OTG B-Peripheral
  - OTG B-device default state if B-side of USB cable is plugged in

- OTG A-Peripheral
  - OTG A-device state after the HNP switches the OTG_HS to its peripheral role

- B-device
  - If the ID line is present, functional and connected to the B-side of the USB cable, and the HNP-capable bit in the Global USB Configuration register (HNPCAP bit in OTG_GUSBCFG) is cleared.

- Peripheral only
  - The force device mode bit (FDMOD) in the Section 62.14.4: OTG USB configuration register (OTG_GUSBCFG) is set to 1, forcing the OTG_HS core to work as an USB peripheral-only. In this case, the ID line is ignored even if it is present on the USB connector.

Note: To build a bus-powered device implementation in case of the B-device or peripheral-only configuration, an external regulator has to be added, that generates the necessary power-supply from VBUS.

62.6.1 SRP-capable peripheral

The SRP capable bit in the Global USB configuration register (SRPCAP bit in OTG_GUSBCFG) enables the OTG_HS to support the session request protocol (SRP). In this way, it allows the remote A-device to save power by switching off VBUS while the USB session is suspended.

The SRP peripheral mode program model is described in detail in the B-device session request protocol section.
62.6.2 Peripheral states

Powered state

The $V_{BUS}$ input detects the B-session valid voltage by which the USB peripheral is allowed to enter the powered state (see USB2.0 section 9.1). The OTG_HS then automatically connects the DP pull-up resistor to signal full-speed device connection to the host and generates the session request interrupt (SRQINT bit in OTG_GINTSTS) to notify the powered state.

The $V_{BUS}$ input also ensures that valid $V_{BUS}$ levels are supplied by the host during USB operations. If a drop in $V_{BUS}$ below B-session valid happens to be detected (for instance because of a power disturbance or if the host port has been switched off), the OTG_HS automatically disconnects and the session end detected (SEDET bit in OTG_GOTGINT) interrupt is generated to notify that the OTG_HS has exited the powered state.

In the powered state, the OTG_HS expects to receive some reset signaling from the host. No other USB operation is possible. When a reset signaling is received the reset detected interrupt (USBRST in OTG_GINTSTS) is generated. When the reset signaling is complete, the enumeration done interrupt (ENUMDNE bit in OTG_GINTSTS) is generated and the OTG_HS enters the Default state.

Soft disconnect

The powered state can be exited by software with the soft disconnect feature. The DP pull-up resistor is removed by setting the soft disconnect bit in the device control register (SDIS bit in OTG_DCTL), causing a device disconnect detection interrupt on the host side even though the USB cable was not really removed from the host port.

Default state

In the Default state the OTG_HS expects to receive a SET_ADDRESS command from the host. No other USB operation is possible. When a valid SET_ADDRESS command is decoded on the USB, the application writes the corresponding number into the device address field in the device configuration register (DAD bit in OTG_DCFG). The OTG_HS then enters the address state and is ready to answer host transactions at the configured USB address.

Suspended state

The OTG_HS peripheral constantly monitors the USB activity. After counting 3 ms of USB idleness, the early suspend interrupt (ESUSP bit in OTG_GINTSTS) is issued, and confirmed 3 ms later, if appropriate, by the suspend interrupt (USBSUSP bit in OTG_GINTSTS). The device suspend bit is then automatically set in the device status register (SUSPSTS bit in OTG_DSTS) and the OTG_HS enters the suspended state.

The suspended state may optionally be exited by the device itself. In this case the application sets the remote wakeup signaling bit in the device control register (RWUSIG bit in OTG_DCTL) and clears it after 1 to 15 ms.

When a resume signaling is detected from the host, the resume interrupt (WKUPINT bit in OTG_GINTSTS) is generated and the device suspend bit is automatically cleared.
62.6.3 Peripheral endpoints

The OTG_HS core instantiates the following USB endpoints:

- **Control endpoint 0:**
  - Bidirectional and handles control messages only
  - Separate set of registers to handle in and out transactions
  - Proper control (OTG_DIEPCTL0/OTG_DOEPCTL0), transfer configuration (OTG_DIEPTSIZ0/OTG_DOEPTSIZ0), and status-interrupt (OTG_DIEPINT0/OTG_DOEPINT0) registers. The available set of bits inside the control and transfer size registers slightly differs from that of other endpoints

- **8 IN endpoints**
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has proper control (OTG_DIEPCTLx), transfer configuration (OTG_DIEPTSIZx), and status-interrupt (OTG_DIEPINTx) registers
  - The device IN endpoints common interrupt mask register (OTG_DIEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the IN endpoints (EP0 included)
  - Support for incomplete isochronous IN transfer interrupt (IISOIXFR bit in OTG_GINTSTS), asserted when there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

- **8 OUT endpoints**
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has a proper control (OTG_DOEPCTLx), transfer configuration (OTG_DOEPTSIZx) and status-interrupt (OTG_DOEPINTx) register
  - Device OUT endpoints common interrupt mask register (OTG_DOEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the OUT endpoints (EP0 included)
  - Support for incomplete isochronous OUT transfer interrupt (INCOMPISOOUT bit in OTG_GINTSTS), asserted when there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).
Endpoint control

- The following endpoint controls are available to the application through the device endpoint-x IN/OUT control register (OTG_DIEPCTLx/OTG_DOEPCTLx):
  - Endpoint enable/disable
  - Endpoint activate in current configuration
  - Program USB transfer type (isochronous, bulk, interrupt)
  - Program supported packet size
  - Program Tx FIFO number associated with the IN endpoint
  - Program the expected or transmitted data0/data1 PID (bulk/interrupt only)
  - Program the even/odd frame during which the transaction is received or transmitted (isochronous only)
  - Optionally program the NAK bit to always negative-acknowledge the host regardless of the FIFO status
  - Optionally program the STALL bit to always stall host tokens to that endpoint
  - Optionally program the SNOOP mode for OUT endpoint not to check the CRC field of received data

Endpoint transfer

The device endpoint-x transfer size registers (OTG_DIEPTSIZx/OTG_DOEPTSIZx) allow the application to program the transfer size parameters and read the transfer status. Programming must be done before setting the endpoint enable bit in the endpoint control register. Once the endpoint is enabled, these fields are read-only as the OTG_HS core updates them with the current transfer status.

The following transfer parameters can be programmed:
- Transfer size in bytes
- Number of packets that constitute the overall transfer size

Endpoint status/interrupt

The device endpoint-x interrupt registers (OTG_DIEPINTx/OTG_DOPEPINTx) indicate the status of an endpoint with respect to USB- and AHB-related events. The application must read these registers when the OUT endpoint interrupt bit or the IN endpoint interrupt bit in the core interrupt register (OEPINT bit in OTG_GINTSTS or IEPINT bit in OTG_GINTSTS, respectively) is set. Before the application can read these registers, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.
The peripheral core provides the following status checks and interrupt generation:
- Transfer completed interrupt, indicating that data transfer was completed on both the application (AHB) and USB sides
- Setup stage has been done (control-out only)
- Associated transmit FIFO is half or completely empty (in endpoints)
- NAK acknowledge has been transmitted to the host (isochronous-in only)
- IN token received when Tx FIFO was empty (bulk-in/interrupt-in only)
- Out token received when endpoint was not yet enabled
- Babble error condition has been detected
- Endpoint disable by application is effective
- Endpoint NAK by application is effective (isochronous-in only)
- More than 3 back-to-back setup packets were received (control-out only)
- Timeout condition detected (control-in only)
- Isochronous out packet has been dropped, without generating an interrupt

62.7 **OTG_HS as a USB host**

This section gives the functional description of the OTG_HS in the USB host mode. The OTG_HS works as a USB host in the following circumstances:
- OTG A-host
  - OTG A-device default state when the A-side of the USB cable is plugged in
- OTG B-host
  - OTG B-device after HNP switching to the host role
- A-device
  - If the ID line is present, functional and connected to the A-side of the USB cable, and the HNP-capable bit is cleared in the Global USB Configuration register (HNPCAP bit in OTG_GUSBCFG). Integrated pull-down resistors are automatically set on the DP/DM lines.
- Host only
  - The force host mode bit (FHMOD) in the OTG USB configuration register (OTG_GUSBCFG) forces the OTG_HS core to work as a USB host-only. In this case, the ID line is ignored even if present on the USB connector. Integrated pull-down resistors are automatically set on the DP/DM lines.

**Note:** *On-chip 5 V VBUS generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch must be added externally to drive the 5 V VBUS line. The external charge pump can be driven by any GPIO output. This is required for the OTG A-host, A-device and host-only configurations.*

62.7.1 **SRP-capable host**

SRP support is available through the SRP capable bit in the global USB configuration register (SRPCAP bit in OTG_GUSBCFG). With the SRP feature enabled, the host can save power by switching off the VBUS power while the USB session is suspended.

The SRP host mode program model is described in detail in the *A-device session request protocol* section.
62.7.2 USB host states

Host port power

On-chip 5 V VBUS generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch, must be added externally to drive the 5 V VBUS line. The external charge pump can be driven by any GPIO output or via an I²C interface connected to an external PMIC (power management IC). When the application decides to power on VBUS, it must also set the port power bit in the host port control and status register (PPWR bit in OTG_HPRT).

VBUS valid

When HNP or SRP is enabled the VBUS sensing pin should be connected to VBUS. The VBUS input ensures that valid VBUS levels are supplied by the charge pump during USB operations. Any unforeseen VBUS voltage drop below the VBUS valid threshold (4.4 V) leads to an OTG interrupt triggered by the session end detected bit (SEDET bit in OTG_GOTGINT). The application is then required to remove the VBUS power and clear the port power bit.

When HNP and SRP are both disabled, the VBUS sensing pin does not need to be connected to VBUS.

The charge pump overcurrent flag can also be used to prevent electrical damage. Connect the overcurrent flag output from the charge pump to any GPIO input and configure it to generate a port interrupt on the active level. The overcurrent ISR must promptly disable the VBUS generation and clear the port power bit.

Host detection of a peripheral connection

If SRP or HNP are enabled, even if USB peripherals or B-devices can be attached at any time, the OTG_HS will not detect any bus connection until VBUS is no longer sensed at a valid level (5 V). When VBUS is at a valid level and a remote B-device is attached, the OTG_HS core issues a host port interrupt triggered by the device connected bit in the host port control and status register (PCDET bit in OTG_HPRT).

When HNP and SRP are both disabled, USB peripherals or B-device are detected as soon as they are connected. The OTG_HS core issues a host port interrupt triggered by the device connected bit in the host port control and status (PCDET bit in OTG_HPRT).

Host detection of peripheral a disconnection

The peripheral disconnection event triggers the disconnect detected interrupt (DISCINT bit in OTG_GINTSTS).

Host enumeration

After detecting a peripheral connection the host must start the enumeration process by sending USB reset and configuration commands to the new peripheral.

Before starting to drive a USB reset, the application waits for the OTG interrupt triggered by the debounce done bit (DBCDNE bit in OTG_GOTGINT), which indicates that the bus is stable again after the electrical debounce caused by the attachment of a pull-up resistor on DP (FS) or DM (LS).

The application drives a USB reset signaling (single-ended zero) over the USB by keeping the port reset bit set in the host port control and status register (PRST bit in OTG_HPRT) for
a minimum of 10 ms and a maximum of 20 ms. The application takes care of the timing count and then of clearing the port reset bit.

Once the USB reset sequence has completed, the host port interrupt is triggered by the port enable/disable change bit (PENCHNG bit in OTG_HPRT). This informs the application that the speed of the enumerated peripheral can be read from the port speed field in the host port control and status register (PSPD bit in OTG_HPRT) and that the host is starting to drive SOFs (FS) or Keep alives (LS). The host is now ready to complete the peripheral enumeration by sending peripheral configuration commands.

**Host suspend**

The application decides to suspend the USB activity by setting the port suspend bit in the host port control and status register (PSUSP bit in OTG_HPRT). The OTG_HS core stops sending SOFs and enters the suspended state.

The suspended state can be optionally exited on the remote device’s initiative (remote wakeup). In this case the remote wakeup interrupt (WKUPINT bit in OTG_GINTSTS) is generated upon detection of a remote wakeup signaling, the port resume bit in the host port control and status register (PRES bit in OTG_HPRT) self-sets, and resume signaling is automatically driven over the USB. The application must time the resume window and then clear the port resume bit to exit the suspended state and restart the SOF.

If the suspended state is exited on the host initiative, the application must set the port resume bit to start resume signaling on the host port, time the resume window and finally clear the port resume bit.

### 62.7.3 Host channels

The OTG_HS core instantiates 16 host channels. Each host channel supports an USB host transfer (USB pipe). The host is not able to support more than 16 transfer requests at the same time. If more than 16 transfer requests are pending from the application, the host controller driver (HCD) must re-allocate channels when they become available from previous duty, that is, after receiving the transfer completed and channel halted interrupts.

Each host channel can be configured to support in/out and any type of periodic/nonperiodic transaction. Each host channel makes us of proper control (OTG_HCCHAR), transfer configuration (OTG_HCTSIZ) and status/interrupt (OTG_HCINT) registers with associated mask (OTG_HCINTMSK) registers.

#### Host channel control

- The following host channel controls are available to the application through the host channel-x characteristics register (OTG_HCCHARx):
  - Channel enable/disable
  - Program the HS/FS/LS speed of target USB peripheral
  - Program the address of target USB peripheral
  - Program the endpoint number of target USB peripheral
  - Program the transfer IN/OUT direction
  - Program the USB transfer type (control, bulk, interrupt, isochronous)
  - Program the maximum packet size (MPS)
  - Program the periodic transfer to be executed during odd/even frames
Host channel transfer

The host channel transfer size registers (OTG_HCTSIZx) allow the application to program the transfer size parameters, and read the transfer status. Programming must be done before setting the channel enable bit in the host channel characteristics register. Once the endpoint is enabled the packet count field is read-only as the OTG_HS core updates it according to the current transfer status.

- The following transfer parameters can be programmed:
  - transfer size in bytes
  - number of packets making up the overall transfer size
  - initial data PID

Host channel status/interrupt

The host channel-x interrupt register (OTG_HCINTx) indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read these register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read these registers, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers. The mask bits for each interrupt source of each channel are also available in the OTG_HCINTMSKx register.

- The host core provides the following status checks and interrupt generation:
  - Transfer completed interrupt, indicating that the data transfer is complete on both the application (AHB) and USB sides
  - Channel has stopped due to transfer completed, USB transaction error or disable command from the application
  - Associated transmit FIFO is half or completely empty (IN endpoints)
  - ACK response received
  - NAK response received
  - STALL response received
  - USB transaction error due to CRC failure, timeout, bit stuff error, false EOP
  - Babble error
  - frame overrun
  - data toggle error

62.7.4 Host scheduler

The host core features a built-in hardware scheduler which is able to autonomously re-order and manage the USB transaction requests posted by the application. At the beginning of each frame the host executes the periodic (isochronous and interrupt) transactions first, followed by the nonperiodic (control and bulk) transactions to achieve the higher level of priority granted to the isochronous and interrupt transfer types by the USB specification.

The host processes the USB transactions through request queues (one for periodic and one for nonperiodic). Each request queue can hold up to 8 entries. Each entry represents a pending transaction request from the application, and holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the
requests are written to the queue determines the sequence of the transactions on the USB interface.

At the beginning of each frame, the host processes the periodic request queue first, followed by the nonperiodic request queue. The host issues an incomplete periodic transfer interrupt (IPXFR bit in OTG_GINTSTS) if an isochronous or interrupt transaction scheduled for the current frame is still pending at the end of the current frame. The OTG_HS core is fully responsible for the management of the periodic and nonperiodic request queues. The periodic transmit FIFO and queue status register (OTG_HPTXSTS) and nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) are read-only registers which can be used by the application to read the status of each request queue. They contain:

- The number of free entries currently available in the periodic (nonperiodic) request queue (8 max)
- Free space currently available in the periodic (nonperiodic) Tx FIFO (out-transactions)
- IN/OUT token, host channel number and other status information.

As request queues can hold a maximum of 8 entries each, the application can push to schedule host transactions in advance with respect to the moment they physically reach the SB for a maximum of 8 pending periodic transactions plus 8 pending non-periodic transactions.

To post a transaction request to the host scheduler (queue) the application must check that there is at least 1 entry available in the periodic (nonperiodic) request queue by reading the PTXQSAV bits in the OTG_HNPTXSTS register or NPTQXSAV bits in the OTG_HNPTXSTS register.

### 62.8 OTG_HS SOF trigger

**Figure 716. SOF connectivity (SOF trigger output to TIM and ITR1 connection)**

The OTG_HS core provides means to monitor, track and configure SOF framing in the host and peripheral, as well as an SOF pulse output connectivity feature.

Such utilities are especially useful for adaptive audio clock generation techniques, where the audio peripheral needs to synchronize to the isochronous stream provided by the PC, or the host needs to trim its framing rate according to the requirements of the audio peripheral.
62.8.1 Host SOFs

In host mode the number of PHY clocks occurring between the generation of two consecutive SOF (HS/FS) or Keep-alive (LS) tokens is programmable in the host frame interval register (HFIR), thus providing application control over the SOF framing period. An interrupt is generated at any start of frame (SOF bit in OTG_GINTSTS). The current frame number and the time remaining until the next SOF are tracked in the host frame number register (HFNUM).

A SOF pulse signal, is generated at any SOF starting token and with a width of 20 HCLK cycles. The SOF pulse is also internally connected to the input trigger of the timer, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

62.8.2 Peripheral SOFs

In device mode, the start of frame interrupt is generated each time an SOF token is received on the USB (SOF bit in OTG_GINTSTS). The corresponding frame number can be read from the device status register (FNSOF bit in OTG_DSTS). A SOF pulse signal with a width of 20 HCLK cycles is also generated. The SOF pulse signal is also internally connected to the TIM input trigger, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

The end of periodic frame interrupt (OTG_GINTSTS/EOPF) is used to notify the application when 80%, 85%, 90% or 95% of the time frame interval elapsed depending on the periodic frame interval field in the device configuration register (PFIVL bit in OTG_DCFG). This feature can be used to determine if all of the isochronous traffic for that frame is complete.

62.9 OTG_HS low-power modes

Table 493 below defines the STM32 low power modes and their compatibility with the OTG.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
<th>USB compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>MCU fully active</td>
<td>Required when USB not in suspend state.</td>
</tr>
<tr>
<td>Sleep</td>
<td>USB suspend exit causes the device to exit Sleep mode. Peripheral registers content is kept.</td>
<td>Available while USB is in suspend state.</td>
</tr>
<tr>
<td>Stop</td>
<td>USB suspend exit causes the device to exit Stop mode. Peripheral registers content is kept(^1).</td>
<td>Available while USB is in suspend state.</td>
</tr>
<tr>
<td>Standby</td>
<td>Powered-down. The peripheral must be reinitialized after exiting Standby mode.</td>
<td>Not compatible with USB applications.</td>
</tr>
</tbody>
</table>

\(^1\) Within Stop mode there are different possible settings. Some restrictions may also exist, please refer to Section 6: Power control (PWR) to understand which (if any) restrictions apply when using OTG.

The following bits and procedures reduce power consumption.
The power consumption of the OTG PHY is controlled by two or three bits in the general core configuration register, depending on OTG revision supported.

- **PHY power down (OTG_GCCFG/PWRDWN)**
  It switches on/off the full-speed transceiver module of the PHY. It must be preliminarily set to allow any USB operation

- **VBUS detection enable (OTG_GCCFG/VBDEN)**
  It switches on/off the VBUS sensing comparators associated with OTG operations

Power reduction techniques are available while in the USB suspended state, when the USB session is not yet valid or the device is disconnected.

- **Stop PHY clock (STPPCLK bit in OTG_PCGCCTL)**
  When setting the stop PHY clock bit in the clock gating control register, most of the 48 MHz clock domain internal to the OTG full-speed core is switched off by clock gating. The dynamic power consumption due to the USB clock switching activity is cut even if the 48 MHz clock input is kept running by the application. Most of the transceiver is also disabled, and only the part in charge of detecting the asynchronous resume or remote wakeup event is kept alive.

- **Gate HCLK (GATEHCLK bit in OTG_PCGCCTL)**
  When setting the Gate HCLK bit in the clock gating control register, most of the system clock domain internal to the OTG_HS core is switched off by clock gating. Only the register read and write interface is kept alive. The dynamic power consumption due to the USB clock switching activity is cut even if the system clock is kept running by the application for other purposes.

- **USB system stop**
  When the OTG_HS is in the USB suspended state, the application may decide to drastically reduce the overall power consumption by a complete shut down of all the clock sources in the system. USB System Stop is activated by first setting the Stop PHY clock bit and then configuring the system deep sleep mode in the power control system module (PWR).

  The OTG_HS core automatically reactivates both system and USB clocks by asynchronous detection of remote wakeup (as an host) or resume (as a device) signaling on the USB.

To save dynamic power, the USB data FIFO is clocked only when accessed by the OTG_HS core.

### 62.10 OTG_HS Dynamic update of the OTG_HFIR register

The USB core embeds a dynamic trimming capability of micro-SOF framing period in host mode allowing to synchronize an external device with the micro-SOF frames.

When the OTG_HFIR register is changed within a current micro-SOF frame, the SOF period correction is applied in the next frame as described in Figure 717.

For a dynamic update, it is required to set RLDCTRL=1.
62.11 **OTG_HS data FIFOs**

The USB system features 4 Kbytes of dedicated RAM with a sophisticated FIFO control mechanism. The packet FIFO controller module in the OTG_HS core organizes RAM space into Tx FIFOs into which the application pushes the data to be temporarily stored before the USB transmission, and into a single Rx FIFO where the data received from the USB are temporarily stored before retrieval (popped) by the application. The number of instructed FIFOs and how these are organized inside the RAM depends on the device’s role. In peripheral mode an additional Tx FIFO is instructed for each active IN endpoint. Any FIFO size is software configured to better meet the application requirements.
62.11.1 Peripheral FIFO architecture

Peripheral Rx FIFO

The OTG peripheral uses a single receive FIFO that receives the data directed to all OUT endpoints. Received packets are stacked back-to-back until free space is available in the Rx FIFO. The status of the received packet (which contains the OUT endpoint destination number, the byte count, the data PID and the validity of the received data) is also stored by the core on top of the data payload. When no more space is available, host transactions are NACKed and an interrupt is received on the addressed endpoint. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it more efficient for the USB peripheral to fill in the receive RAM buffer:

- All OUT endpoints share the same RAM buffer (shared FIFO)
- The OTG_HS core can fill in the receive FIFO up to the limit for any host sequence of OUT tokens

The application keeps receiving the Rx FIFO non-empty interrupt (RXFLVL bit in OTG_GINTSTS) as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register (OTG_GRXSTSP) and finally pops data off the receive FIFO by reading from the endpoint-related pop address.
Peripheral Tx FIFOs

The core has a dedicated FIFO for each IN endpoint. The application configures FIFO sizes by writing the endpoint 0 transmit FIFO size register (OTG_DIEPTXF0) for IN endpoint0 and the device IN endpoint transmit FIFOx registers (OTG_DIEPTXFx) for IN endpoint-x.

62.11.2 Host FIFO architecture

Figure 719. Host-mode FIFO address mapping and AHB FIFO access mapping

Host Rx FIFO

The host uses one receiver FIFO for all periodic and nonperiodic transactions. The FIFO is used as a receive buffer to hold the received data (payload of the received packet) from the USB until it is transferred to the system memory. Packets received from any remote IN endpoint are stacked back-to-back until free space is available. The status of each received packet with the host channel destination, byte count, data PID and validity of the received data are also stored into the FIFO. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it highly efficient for the USB host to fill in the receive data buffer:
- All IN configured host channels share the same RAM buffer (shared FIFO)
- The OTG_HS core can fill in the receive FIFO up to the limit for any sequence of IN tokens driven by the host software

The application receives the Rx FIFO not-empty interrupt as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register and finally pops the data off the receive FIFO.
Host Tx FIFOs

The host uses one transmit FIFO for all non-periodic (control and bulk) OUT transactions and one transmit FIFO for all periodic (isochronous and interrupt) OUT transactions. FIFOs are used as transmit buffers to hold the data (payload of the transmit packet) to be transmitted over the USB. The size of the periodic (nonperiodic) Tx FIFO is configured in the host periodic (nonperiodic) transmit FIFO size OTG_HPTXFSIZ / OTG_HNPTXFSIZ) register.

The two Tx FIFO implementation derives from the higher priority granted to the periodic type of traffic over the USB frame. At the beginning of each frame, the built-in host scheduler processes the periodic request queue first, followed by the nonperiodic request queue.

The two transmit FIFO architecture provides the USB host with separate optimization for periodic and nonperiodic transmit data buffer management:

- All host channels configured to support periodic (nonperiodic) transactions in the OUT direction share the same RAM buffer (shared FIFOs)
- The OTG_HS core can fill in the periodic (nonperiodic) transmit FIFO up to the limit for any sequence of OUT tokens driven by the host software

The OTG_HS core issues the periodic Tx FIFO empty interrupt (PTXFE bit in OTG_GINTSTS) as long as the periodic Tx FIFO is half or completely empty, depending on the value of the periodic Tx FIFO empty level bit in the AHB configuration register (PTXFELVL bit in OTG_GAHBCFG). The application can push the transmission data in advance as long as free space is available in both the periodic Tx FIFO and the periodic request queue. The host periodic transmit FIFO and queue status register (OTG_HPTXSTS) can be read to know how much space is available in both.

The OTG_HS core issues the non periodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) as long as the nonperiodic Tx FIFO is half or completely empty depending on the non periodic Tx FIFO empty level bit in the AHB configuration register (TXFELVL bit in OTG_GAHBCFG). The application can push the transmission data as long as free space is available in both the nonperiodic Tx FIFO and nonperiodic request queue. The host nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) can be read to know how much space is available in both.

62.11.3 FIFO RAM allocation

Device mode

Receive FIFO RAM allocation: the application should allocate RAM for SETUP packets:

- 10 locations must be reserved in the receive FIFO to receive SETUP packets on control endpoint. The core does not use these locations, which are reserved for SETUP packets, to write any other data.
- One location is to be allocated for Global OUT NAK.
- Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous endpoints are enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.
- Along with the last packet for each endpoint, transfer complete status information is also pushed to the FIFO. One location for each OUT endpoint is recommended.
Device RxFIFO =

\( (5 \times \text{number of control endpoints} + 8) + ((\text{largest USB packet used / 4}) + 1 \text{ for status information}) + (2 \times \text{number of OUT endpoints}) + 1 \text{ for Global NAK} \)

Example: The MPS is 1,024 bytes for a periodic USB packet and 512 bytes for a non-periodic USB packet. There are three OUT endpoints, three IN endpoints, one control endpoint, and three host channels.

Device RxFIFO = \((5 \times 1 + 8) + ((1,024 / 4) +1) + (2 \times 4) + 1 = 279 \)

Transmit FIFO RAM allocation: the minimum RAM space required for each IN endpoint Transmit FIFO is the maximum packet size for that particular IN endpoint.

Note: More space allocated in the transmit IN endpoint FIFO results in better performance on the USB.

**Host mode**

Receive FIFO RAM allocation:

Status information is written to the FIFO along with each received packet. Therefore, a minimum space of \((\text{largest packet size / 4}) + 1\) must be allocated to receive packets. If multiple isochronous channels are enabled, then at least two \((\text{largest packet size / 4}) + 1\) spaces must be allocated to receive back-to-back packets. Typically, two \((\text{largest packet size / 4}) + 1\) spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.

Along with the last packet in the host channel, transfer complete status information is also pushed to the FIFO. So one location must be allocated for this.

Host RxFIFO = \((\text{largest USB packet used / 4}) + 1 \text{ for status information} + 1 \text{ transfer complete}\)

Example: Host RxFIFO = \(((1,024 / 4) + 1) + 1 = 258 \)

Transmit FIFO RAM allocation:

The minimum amount of RAM required for the host Non-periodic Transmit FIFO is the largest maximum packet size among all supported non-periodic OUT channels.

Typically, two largest packet sizes worth of space is recommended, so that when the current packet is under transfer to the USB, the CPU can get the next packet.

Non-Periodic TxFIFO = \(\text{largest non-periodic USB packet used / 4}\)

Example: Non-Periodic TxFIFO = \((512 / 4) = 128 \)

The minimum amount of RAM required for host periodic Transmit FIFO is the largest maximum packet size out of all the supported periodic OUT channels. If there is at least one isochronous OUT endpoint, then the space must be at least two times the maximum packet size of that channel.

Host Periodic TxFIFO = \(\text{largest periodic USB packet used / 4}\)

Example: Host Periodic TxFIFO = \(1,024 / 4 = 256 \)

Note: More space allocated in the Transmit Non-periodic FIFO results in better performance on the USB.
62.12 OTG_HS interrupts

When the OTG_HS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

*Figure 720* shows the interrupt hierarchy.
1. OTG_HS_WKUP becomes active (high state) when resume condition occurs during L1 SLEEP or L2 SUSPEND states.
62.13 OTG_HS control and status registers

By reading from and writing to the control and status registers (CSRs) through the AHB slave interface, the application controls the OTG_HS controller. These registers are 32 bits wide, and the addresses are 32-bit block aligned. The OTG_HS registers must be accessed by words (32 bits).

CSRs are classified as follows:
- Core global registers
- Host-mode registers
- Host global registers
- Host port CSRs
- Host channel-specific registers
- Device-mode registers
- Device global registers
- Device endpoint-specific registers
- Power and clock-gating registers
- Data FIFO (DFIFO) access registers

Only the core global, power and clock-gating, data FIFO access, and host port control and status registers can be accessed in both host and device modes. When the OTG_HS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

62.13.1 CSR memory map

The host and device mode registers occupy different addresses. All registers are implemented in the AHB clock domain.

Global CSR map

These registers are available in both host and device modes.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Address offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_GOTGCTL</td>
<td>0x000</td>
<td>Section 62.14.1: OTG control and status register (OTG_GOTGCTL)</td>
</tr>
<tr>
<td>OTG_GOTGIN</td>
<td>0x004</td>
<td>Section 62.14.2: OTG interrupt register (OTG_GOTGIN)</td>
</tr>
<tr>
<td>OTG_GAHBCFG</td>
<td>0x008</td>
<td>Section 62.14.3: OTG AHB configuration register (OTG_GAHBCFG)</td>
</tr>
<tr>
<td>OTG_GUSBCFG</td>
<td>0x00C</td>
<td>Section 62.14.4: OTG USB configuration register (OTG_GUSBCFG)</td>
</tr>
<tr>
<td>OTG_GRSTCTL</td>
<td>0x010</td>
<td>Section 62.14.5: OTG reset register (OTG_GRSTCTL)</td>
</tr>
<tr>
<td>OTG_GINTSTS</td>
<td>0x014</td>
<td>Section 62.14.6: OTG core interrupt register (OTG_GINTSTS)</td>
</tr>
<tr>
<td>OTG_GINTMSK</td>
<td>0x018</td>
<td>Section 62.14.7: OTG interrupt mask register (OTG_GINTMSK)</td>
</tr>
</tbody>
</table>
### Table 494. Core global control and status registers (CSRs) (continued)

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Address offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_GRXSTSR</td>
<td>0x01C</td>
<td>Section 62.14.8: OTG receive status debug read register (OTG_GRXSTSR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 62.14.9: OTG receive status debug read [alternate] (OTG_GRXSTSR)</td>
</tr>
<tr>
<td>OTG_GRXSTSP</td>
<td>0x020</td>
<td>Section 62.14.10: OTG status read and pop registers (OTG_GRXSTSP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 62.14.11: OTG status read and pop registers [alternate] (OTG_GRXSTSP)</td>
</tr>
<tr>
<td>OTG_GRXFSIZ</td>
<td>0x024</td>
<td>Section 62.14.12: OTG receive FIFO size register (OTG_GRXFSIZ)</td>
</tr>
<tr>
<td>OTG_HNPTXF(SIZ/</td>
<td>0x028</td>
<td>Section 62.14.13: OTG host non-periodic transmit FIFO size register (OTG_HNPTXF(SIZ/Endpoint 0 Transmit FIFO size (OTG_DIEPTXF0))</td>
</tr>
<tr>
<td>OTG_DIEPTXF0(1)</td>
<td></td>
<td>Section 62.14.14: OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)</td>
</tr>
<tr>
<td>OTG_GCCFG</td>
<td>0x038</td>
<td>Section 62.14.15: OTG general core configuration register (OTG_GCCFG)</td>
</tr>
<tr>
<td>OTG_CID</td>
<td>0x03C</td>
<td>Section 62.14.16: OTG core ID register (OTG_CID)</td>
</tr>
<tr>
<td>OTG_GLPMCFG</td>
<td>0x54</td>
<td>Section 62.14.17: OTG core LPM configuration register (OTG_GLPMCFG)</td>
</tr>
<tr>
<td>OTG_HPTXFSIZ</td>
<td>0x100</td>
<td>Section 62.14.18: OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)</td>
</tr>
<tr>
<td>OTG_DIEPTXFx</td>
<td>0x104, 0x108, 0x120</td>
<td>Section 62.14.19: OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)</td>
</tr>
</tbody>
</table>

1. The general rule is to use OTG_HNPTXF(SIZ for host mode and OTG_DIEPTXF0 for device mode.

### Host-mode CSR map

These registers must be programmed every time the core changes to host mode.

### Table 495. Host-mode control and status registers (CSRs)

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_HCFG</td>
<td>0x400</td>
<td>Section 62.14.21: OTG host configuration register (OTG_HCFG)</td>
</tr>
<tr>
<td>OTG_HFIR</td>
<td>0x404</td>
<td>Section 62.14.22: OTG host frame interval register (OTG_HFIR)</td>
</tr>
<tr>
<td>OTG_HFNUM</td>
<td>0x408</td>
<td>Section 62.14.23: OTG host frame number/frame time remaining register (OTG_HFNUM)</td>
</tr>
<tr>
<td>OTG_HPTXSTS</td>
<td>0x410</td>
<td>Section 62.14.24: OTG Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)</td>
</tr>
<tr>
<td>OTG_HAINT</td>
<td>0x414</td>
<td>Section 62.14.25: OTG host all channels interrupt register (OTG_HAINT)</td>
</tr>
<tr>
<td>OTG_HAINTMSK</td>
<td>0x418</td>
<td>Section 62.14.26: OTG host all channels interrupt mask register (OTG_HAINTMSK)</td>
</tr>
</tbody>
</table>
### Table 495. Host-mode control and status registers (CSRs) (continued)

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_HFLBADDR</td>
<td>0x41C</td>
<td>Section 62.14.27: OTG host frame list base address register (OTG_HFLBADDR)</td>
</tr>
<tr>
<td>OTG_HPRT</td>
<td>0x440</td>
<td>Section 62.14.28: OTG host port control and status register (OTG_HPRT)</td>
</tr>
<tr>
<td>OTG_HCCHARx</td>
<td></td>
<td>Section 62.14.29: OTG host channel x characteristics register (OTG_HCCHARx)</td>
</tr>
<tr>
<td>OTG_HCSPLTx</td>
<td>0x504 0x524</td>
<td>Section 62.14.30: OTG host channel x split control register (OTG_HCSPLTx)</td>
</tr>
<tr>
<td>OTG_HCINTx</td>
<td>0x508 0x528</td>
<td>Section 62.14.31: OTG host channel x interrupt register (OTG_HCINTx)</td>
</tr>
<tr>
<td>OTG_HCSPLTx</td>
<td>0x500 0x520</td>
<td>Section 62.14.30: OTG host channel x split control register (OTG_HCSPLTx)</td>
</tr>
<tr>
<td>OTG_HCINTx</td>
<td>0x508 0x528</td>
<td>Section 62.14.31: OTG host channel x interrupt register (OTG_HCINTx)</td>
</tr>
<tr>
<td>OTG_HCINTMSKx</td>
<td>0x50C 0x52C</td>
<td>Section 62.14.32: OTG host channel x interrupt mask register (OTG_HCINTMSKx)</td>
</tr>
<tr>
<td>OTG_HCINTMSKx</td>
<td></td>
<td>Section 62.14.32: OTG host channel x interrupt mask register (OTG_HCINTMSKx)</td>
</tr>
<tr>
<td>OTG_HCTSIZx</td>
<td>0x510 0x530</td>
<td>Section 62.14.33: OTG host channel x transfer size register (OTG_HCTSIZx)</td>
</tr>
<tr>
<td>OTG_HCTSIZSGx</td>
<td></td>
<td>Section 62.14.34: OTG host channel x transfer size register (OTG_HCTSIZSGx)</td>
</tr>
<tr>
<td>OTG_HCDMAx</td>
<td>0x514 0x534</td>
<td>Section 62.14.35: OTG host channel x DMA address register in buffer DMA [alternate] (OTG_HCDMAx)</td>
</tr>
<tr>
<td>OTG_HCDMASGx</td>
<td>0x514 0x534</td>
<td>Section 62.14.36: OTG host channel x DMA address register in scatter/gather DMA [alternate] (OTG_HCDMASGx)</td>
</tr>
<tr>
<td>OTG_HCDMABx</td>
<td>0x51C 0x53C</td>
<td>Section 62.14.37: OTG host channel-n DMA address buffer register (OTG_HCDMABx)</td>
</tr>
</tbody>
</table>
Device-mode CSR map

These registers must be programmed every time the core changes to device mode.

**Table 496. Device-mode control and status registers**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_DCFG</td>
<td>0x800</td>
<td>Section 62.14.39: OTG device configuration register (OTG_DCFG)</td>
</tr>
<tr>
<td>OTG_DCTL</td>
<td>0x804</td>
<td>Section 62.14.40: OTG device control register (OTG_DCTL)</td>
</tr>
<tr>
<td>OTG_DSTS</td>
<td>0x808</td>
<td>Section 62.14.41: OTG device status register (OTG_DSTS)</td>
</tr>
<tr>
<td>OTG_DIEPMSK</td>
<td>0x810</td>
<td>Section 62.14.42: OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)</td>
</tr>
<tr>
<td>OTG_DOEPMmask</td>
<td>0x814</td>
<td>Section 62.14.43: OTG device OUT endpoint common interrupt mask register (OTG_DOEPMmask)</td>
</tr>
<tr>
<td>OTG_DAINT</td>
<td>0x818</td>
<td>Section 62.14.44: OTG device all endpoints interrupt register (OTG_DAINT)</td>
</tr>
<tr>
<td>OTG_DAINTMSK</td>
<td>0x81C</td>
<td>Section 62.14.45: OTG all endpoints interrupt mask register (OTG_DAINTMSK)</td>
</tr>
<tr>
<td>OTG_DVBUSDIS</td>
<td>0x828</td>
<td>Section 62.14.46: OTG device VBUS discharge time register (OTG_DVBUSDIS)</td>
</tr>
<tr>
<td>OTG_DVBUSPULSE</td>
<td>0x82C</td>
<td>Section 62.14.47: OTG device VBUS pulsing time register (OTG_DVBUSPULSE)</td>
</tr>
<tr>
<td>OTG_DTHRCTL</td>
<td>0x830</td>
<td>Section 62.14.48: OTG device threshold control register (OTG_DTHRCTL)</td>
</tr>
<tr>
<td>OTG_DIEPEMPMSK</td>
<td>0x834</td>
<td>Section 62.14.49: OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)</td>
</tr>
<tr>
<td>OTG_DEACHINT</td>
<td>0x838</td>
<td>Section 62.14.50: OTG device each endpoint interrupt register (OTG_DEACHINT)</td>
</tr>
<tr>
<td>OTG_DEACHINTMSK</td>
<td>0x83C</td>
<td>Section 62.14.51: OTG device each endpoint interrupt mask register (OTG_DEACHINTMSK)</td>
</tr>
<tr>
<td>OTG_HS_DIEPEACHM SK1</td>
<td>0x844</td>
<td>Section 62.14.52: OTG device each IN endpoint-1 interrupt mask register (OTG_HS_DIEPEACHM SK1)</td>
</tr>
<tr>
<td>OTG_HS_DOEPEACHM SK1</td>
<td>0x884</td>
<td>Section 62.14.53: OTG device each OUT endpoint-1 interrupt mask register (OTG_HS_DOEPEACHM SK1)</td>
</tr>
<tr>
<td>OTG_DIEPCTLx</td>
<td>0x900, 0x920, 0xA00</td>
<td>Section 62.14.54: OTG device IN endpoint x control register (OTG_DIEPCTLx)</td>
</tr>
<tr>
<td>OTG_DIEPINTx</td>
<td>0x908, 0x928, 0x9E8</td>
<td>Section 62.14.55: OTG device IN endpoint x interrupt register (OTG_DIEPINTx)</td>
</tr>
</tbody>
</table>
### Data FIFO (DFIFO) access register map

These registers, available in both host and device modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_DIEPTSIZ0</td>
<td>0x910</td>
<td>Section 62.14.56: OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)</td>
</tr>
<tr>
<td>OTG_DIEPDMAx</td>
<td>0x914, 0x934, 0x9F4</td>
<td>Section 62.14.57: OTG device IN endpoint x DMA address register (OTG_DIEPDMAx)</td>
</tr>
<tr>
<td>OTG_DTXFSTSx</td>
<td>0x918, 0x938, 0xF8</td>
<td>Section 62.14.58: OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)</td>
</tr>
<tr>
<td>OTG_DIEPTSIZx</td>
<td>0x930, 0x950, 0xF0</td>
<td>Section 62.14.59: OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)</td>
</tr>
<tr>
<td>OTG_DOEPCTL0</td>
<td>0xB00</td>
<td>Section 62.14.60: OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)</td>
</tr>
<tr>
<td>OTG_DOEPIHTx</td>
<td>0xB08, 0xB28, 0xC08</td>
<td>Section 62.14.61: OTG device OUT endpoint x interrupt register (OTG_DOEPIHTx)</td>
</tr>
<tr>
<td>OTG_DOEPTSIZ0</td>
<td>0xB10</td>
<td>Section 62.14.62: OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)</td>
</tr>
<tr>
<td>OTG_DOEPDMAx</td>
<td>0xB14, 0xB34, 0xC14</td>
<td>Section 62.14.63: OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx)</td>
</tr>
<tr>
<td>OTG_DOEPCTLx</td>
<td>0xB20, 0xB40, 0xC00</td>
<td>Section 62.14.64: OTG device OUT endpoint x control register (OTG_DOEPCTLx)</td>
</tr>
<tr>
<td>OTG_DOEPTSIZx</td>
<td>0xB30, 0xB50, 0xBF0</td>
<td>Section 62.14.65: OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)</td>
</tr>
</tbody>
</table>
Power and clock gating CSR map

There is a single register for power and clock gating. It is available in both host and device modes.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_PCGCCTL</td>
<td>0xE00–0xE04</td>
<td>Section 62.14.66: OTG power and clock gating control register (OTG_PCGCCTL)</td>
</tr>
</tbody>
</table>

### 62.14 OTG_HS registers

These registers are available in both host and device modes, and do not need to be reprogrammed when switching between these modes.

Bit values in the register descriptions are expressed in binary unless otherwise specified.

#### 62.14.1 OTG control and status register (OTG_GOTGCTL)

Address offset: 0x000

Reset value: 0x0001 0000

The OTG_GOTGCTL register controls the behavior and reflects the status of the OTG function of the core.
Bits 31:22  Reserved, must be kept at reset value.

Bit 21  **CURMOD**: Current mode of operation
Indicates the current mode (host or device).
0: Device mode
1: Host mode

Bit 20  **OTGVER**: OTG version
Selects the OTG revision.
0: OTG Version 1.3. OTG1.3 is obsolete for new product development.
1: OTG Version 2.0. In this version the core supports only data line pulsing for SRP.

Bit 19  **BSVLD**: B-session valid
Indicates the device mode transceiver status.
0: B-session is not valid.
1: B-session is valid.
In OTG mode, the user can use this bit to determine if the device is connected or disconnected.
*Note: Only accessible in device mode.*

Bit 18  **ASVLD**: A-session valid
Indicates the host mode transceiver status.
0: A-session is not valid
1: A-session is valid
*Note: Only accessible in host mode.*

Bit 17  **DBCT**: Long/short debounce time
Indicates the debounce time of a detected connection.
0: Long debounce time, used for physical connections (100 ms + 2.5 µs)
1: Short debounce time, used for soft connections (2.5 µs)
*Note: Only accessible in host mode.*

Bit 16  **CIDSTS**: Connector ID status
Indicates the connector ID status on a connect event.
0: The OTG_HS controller is in A-device mode
1: The OTG_HS controller is in B-device mode
*Note: Accessible in both device and host modes.*

Bits 15:13  Reserved, must be kept at reset value.

Bit 12  **EHEN**: Embedded host enable
It is used to select between OTG A device state machine and embedded host state machine.
0: OTG A device state machine is selected
1: Embedded host state machine is selected

Bit 11  **DHNPEN**: Device HNP enabled
The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host.
0: HNP is not enabled in the application
1: HNP is enabled in the application
*Note: Only accessible in device mode.*
Bit 10  **HSHNPEN**: host set HNP enable
    The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device.
    0: Host Set HNP is not enabled
    1: Host Set HNP is enabled
    *Note*: *Only accessible in host mode.*

Bit 9  **HNPRQ**: HNP request
    The application sets this bit to initiate an HNP request to the connected USB host. The application can clear this bit by writing a 0 when the host negotiation success status change bit in the OTG_GOTGINT register (HNSSCHG bit in OTG_GOTGINT) is set. The core clears this bit when the HNSSCHG bit is cleared.
    0: No HNP request
    1: HNP request
    *Note*: *Only accessible in device mode.*

Bit 8  **HNGSCS**: Host negotiation success
    The core sets this bit when host negotiation is successful. The core clears this bit when the HNP request (HNPRQ) bit in this register is set.
    0: Host negotiation failure
    1: Host negotiation success
    *Note*: *Only accessible in device mode.*

Bit 7  **BVALOVAL**: B-peripheral session valid override value.
    This bit is used to set override value for Bvalid signal when BVALOEN bit is set.
    0: Bvalid value is '0' when BVALOEN = 1
    1: Bvalid value is '1' when BVALOEN = 1
    *Note*: *Only accessible in device mode.*

Bit 6  **BVALOEN**: B-peripheral session valid override enable.
    This bit is used to enable/disable the software to override the Bvalid signal using the BVALOVAL bit.
    0: Override is disabled and Bvalid signal from the respective PHY selected is used internally by the core
    1: Internally Bvalid received from the PHY is overridden with BVALOVAL bit value
    *Note*: *Only accessible in device mode.*

Bit 5  **AVALOVAL**: A-peripheral session valid override value.
    This bit is used to set override value for Avalid signal when AVALOEN bit is set.
    0: Avalid value is '0' when AVALOEN = 1
    1: Avalid value is '1' when AVALOEN = 1
    *Note*: *Only accessible in device mode.*

Bit 4  **AVALOEN**: A-peripheral session valid override enable.
    This bit is used to enable/disable the software to override the Avalid signal using the AVALOVAL bit.
    0: Override is disabled and Avalid signal from the respective PHY selected is used internally by the core
    1: Internally Avalid received from the PHY is overridden with AVALOVAL bit value
    *Note*: *Only accessible in host mode.*
Bit 3 **VBVALOVAL**: V/bus valid override value.
This bit is used to set override value for vbusvalid signal when VBVALOEN bit is set.
0: vbusvalid value is '0' when VBVALOEN = 1
1: vbusvalid value is '1' when VBVALOEN = 1
*Note: Only accessible in host mode.*

Bit 2 **VBVALOEN**: Vbus valid override enable.
This bit is used to enable/disable the software to override the vbusvalid signal using the VBVALOVAL bit.
0: Override is disabled and vbusvalid signal from the respective PHY selected is used internally by the core
1: Internally vbusvalid received from the PHY is overridden with VBVALOVAL bit value
*Note: Only accessible in host mode.*

Bit 1 **SRQ**: Session request
The application sets this bit to initiate a session request on the USB. The application can clear this bit by writing a 0 when the host negotiation success status change bit in the OTG_GOTGINT register (HNSSCHG bit in OTG_GOTGINT) is set. The core clears this bit when the HNSSCHG bit is cleared.
If the user uses the USB 1.1 full-speed serial transceiver interface to initiate the session request, the application must wait until VBUS discharges to 0.2 V, after the B-session valid bit in this register (BSVLD bit in OTG_GOTGCTL) is cleared.
0: No session request
1: Session request
*Note: Only accessible in device mode.*

Bit 0 **SRQSCS**: Session request success
The core sets this bit when a session request initiation is successful.
0: Session request failure
1: Session request success
*Note: Only accessible in device mode.*

### 62.14.2 OTG interrupt register (OTG_GOTGINT)

Address offset: 0x04
Reset value: 0x0000 0000
The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.
Bits 31:20  Reserved, must be kept at reset value.

Bit 19  **DBCDNE**: Debounce done
The core sets this bit when the debounce is completed after the device connect. The application can start driving USB reset after seeing this interrupt. This bit is only valid when the HNP Capable or SRP Capable bit is set in the OTG_GUSBCFG register (HNPCAP bit or SRPCAP bit in OTG_GUSBCFG, respectively).

*Note:* Only accessible in host mode.

Bit 18  **ADTOCHG**: A-device timeout change
The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect.

*Note:* Accessible in both device and host modes.

Bit 17  **HNNGDET**: Host negotiation detected
The core sets this bit when it detects a host negotiation request on the USB.

*Note:* Accessible in both device and host modes.

Bits 16:10  Reserved, must be kept at reset value.

Bit 9  **HNSSCHG**: Host negotiation success status change
The core sets this bit on the success or failure of a USB host negotiation request. The application must read the host negotiation success bit of the OTG_GOTGCTL register (HNGSCS bit in OTG_GOTGCTL) to check for success or failure.

*Note:* Accessible in both device and host modes.

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

Bit 8  **SRSSCHG**: Session request success status change
The core sets this bit on the success or failure of a session request. The application must read the session request success bit in the OTG_GOTGCTL register (SRQSCS bit in OTG_GOTGCTL) to check for success or failure.

*Note:* Accessible in both device and host modes.

Bit 2  **SEDET**: Session end detected
The core sets this bit to indicate that the level of the voltage on VBUS is no longer valid for a B-Peripheral session when VBUS < 0.8 V.

*Note:* Accessible in both device and host modes.

Bits 1:0  Reserved, must be kept at reset value.
62.14.3 OTG AHB configuration register (OTG_GAHBCFG)

Address offset: 0x008
Reset value: 0x0000 0000

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 PTXFE LVL: Periodic Tx FIFO empty level
Indicates when the periodic Tx FIFO empty interrupt bit in the OTG_GINTSTS register (PTXFE bit in OTG_GINTSTS) is triggered.
0: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is half empty
1: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is completely empty

Note: Only accessible in host mode.

Bit 7 TXFE LVL: Tx FIFO empty level
In device mode, this bit indicates when IN endpoint Transmit FIFO empty interrupt (TXFE in OTG_DIEPINTx) is triggered:
0: The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is half empty
1: The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is completely empty
In host mode, this bit indicates when the nonperiodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) is triggered:
0: The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is half empty
1: The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is completely empty

Bit 6 Reserved, must be kept at reset value.
Bit 5 **DMAEN**: DMA enabled
0: The core operates in slave mode
1: The core operates in DMA mode

Bits 4:1 **HBSTLEN[3:0]**: Burst length/type
0000 Single: Bus transactions use single 32-bit accesses (not recommended)
0001 INCR: Bus transactions use unspecified length accesses (not recommended, uses the INCR AHB bus command)
0011 INCR4: Bus transactions target 4x 32-bit accesses
0101 INCR8: Bus transactions target 8x 32-bit accesses
0111 INCR16: Bus transactions based on 16x 32-bit accesses
Others: Reserved

Bit 0 **GINTMSK**: Global interrupt mask
The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bit’s setting, the interrupt status registers are updated by the core.
0: Mask the interrupt assertion to the application.
1: Unmask the interrupt assertion to the application.

Note: Accessible in both device and host modes.

### 62.14.4 OTG USB configuration register (OTG_GUSBCFG)

Address offset: 0x00C
Reset value: 0x0000 1400

This register can be used to configure the core after power-on or a changing to host mode or device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:29</td>
<td>FD MOD, FH MOD, Res.</td>
</tr>
<tr>
<td>28:26</td>
<td>Res, Res, ULPI IPD, PTCI</td>
</tr>
<tr>
<td>25:24</td>
<td>Res, ULPI PTCI</td>
</tr>
<tr>
<td>23</td>
<td>Res</td>
</tr>
<tr>
<td>22</td>
<td>Res</td>
</tr>
<tr>
<td>21:19</td>
<td>Res, Res, ULPI TSDPS</td>
</tr>
<tr>
<td>18</td>
<td>Res</td>
</tr>
<tr>
<td>17</td>
<td>Res</td>
</tr>
<tr>
<td>16</td>
<td>Res, ULPI VBUSI</td>
</tr>
<tr>
<td>15</td>
<td>Res</td>
</tr>
<tr>
<td>14</td>
<td>Res</td>
</tr>
<tr>
<td>13</td>
<td>Res</td>
</tr>
<tr>
<td>12</td>
<td>Res</td>
</tr>
<tr>
<td>11</td>
<td>Res</td>
</tr>
<tr>
<td>10</td>
<td>Res</td>
</tr>
<tr>
<td>9</td>
<td>Res</td>
</tr>
<tr>
<td>8</td>
<td>Res</td>
</tr>
<tr>
<td>7</td>
<td>Res</td>
</tr>
<tr>
<td>6</td>
<td>Res</td>
</tr>
<tr>
<td>5</td>
<td>Res</td>
</tr>
<tr>
<td>4</td>
<td>Res</td>
</tr>
<tr>
<td>3:2</td>
<td>TOCAL[2:0]</td>
</tr>
<tr>
<td>1</td>
<td>Res</td>
</tr>
<tr>
<td>0</td>
<td>Res</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bit 31: Reserved, must be kept at reset value.

Bit 30  **FDMOD**: Force device mode
- Writing a 1 to this bit, forces the core to device mode irrespective of the OTG_ID input pin.
  - 0: Normal mode
  - 1: Force device mode
- After setting the force bit, the application must wait at least 25 ms before the change takes effect.
  
  *Note*: **Accessible in both device and host modes.**

Bit 29  **FHMOD**: Force host mode
- Writing a 1 to this bit, forces the core to host mode irrespective of the OTG_ID input pin.
  - 0: Normal mode
  - 1: Force host mode
- After setting the force bit, the application must wait at least 25 ms before the change takes effect.
  
  *Note*: **Accessible in both device and host modes.**

Bits 28:26: Reserved, must be kept at reset value.

Bit 25  **ULPIIPD**: ULPI interface protect disable
- This bit controls the circuitry built in the PHY to protect the ULPI interface when the link tri-states stp and data. Any pull-up or pull-down resistors employed by this feature can be disabled. Refer to the ULPI specification for more details.
  - 0: Enables the interface protection circuit
  - 1: Disables the interface protection circuit

Bit 24  **PTCI**: Indicator pass through
- This bit controls whether the complement output is qualified with the internal $V_{BUS}$ valid comparator before being used in the $V_{BUS}$ state in the RX CMD. Refer to the ULPI specification for more details.
  - 0: Complement Output signal is qualified with the Internal $V_{BUS}$ valid comparator
  - 1: Complement Output signal is not qualified with the Internal $V_{BUS}$ valid comparator

Bit 23  **PCCI**: Indicator complement
- This bit controls the PHY to invert the ExternalVbusIndicator input signal, and generate the complement output. Refer to the ULPI specification for more details.
  - 0: PHY does not invert the ExternalVbusIndicator signal
  - 1: PHY inverts ExternalVbusIndicator signal

Bit 22  **TSDPS**: TermSel DLine pulsing selection
- This bit selects utmi_termselect to drive the data line pulse during SRP (session request protocol).
  - 0: Data line pulsing using utmi_txvalid (default)
  - 1: Data line pulsing using utmi_termsel

Bit 21  **ULPIEVBUSI**: ULPI external $V_{BUS}$ indicator
- This bit indicates to the ULPI PHY to use an external $V_{BUS}$ overcurrent indicator.
  - 0: PHY uses an internal $V_{BUS}$ valid comparator
  - 1: PHY uses an external $V_{BUS}$ valid comparator

Bit 20  **ULPIEVBUSD**: ULPI External $V_{BUS}$ Drive
- This bit selects between internal or external supply to drive 5 V on $V_{BUS}$ in the ULPI PHY.
  - 0: PHY drives $V_{BUS}$ using internal charge pump (default)
  - 1: PHY drives $V_{BUS}$ using external supply.
Bit 19 **ULPICSM**: ULPI clock SuspendM
This bit sets the ClockSuspendM bit in the interface control register on the ULPI PHY. This bit applies only in the serial and car kit modes.
- 0: PHY powers down the internal clock during suspend
- 1: PHY does not power down the internal clock

Bit 18 **ULPIAR**: ULPI Auto-resume
This bit sets the AutoResume bit in the interface control register on the ULPI PHY.
- 0: PHY does not use AutoResume feature
- 1: PHY uses AutoResume feature

Bit 17 **ULPIFSLS**: ULPI FS/LS select
The application uses this bit to select the FS/LS serial interface for the ULPI PHY. This bit is valid only when the FS serial transceiver is selected on the ULPI PHY.
- 0: ULPI interface
- 1: ULPI FS/LS serial interface

Bit 16 Reserved, must be kept at reset value.

Bit 15 **PHYLPC**: PHY Low-power clock select
This bit selects either 480 MHz or 48 MHz (low-power) PHY mode. In FS and LS modes, the PHY can usually operate on a 48 MHz clock to save power.
- 0: 480 MHz internal PLL clock
- 1: 48 MHz external clock

In 480 MHz mode, the UTMI interface operates at either 60 or 30 MHz, depending on whether the 8- or 16-bit data width is selected. In 48 MHz mode, the UTMI interface operates at 48 MHz in FS and LS modes.

Bit 14 Reserved, must be kept at reset value.

Bits 13:10 **TRDT[3:0]**: USB turnaround time
These bits allow to set the turnaround time in PHY clocks. They must be configured according to or Table 500: TRDT values (HS), depending on the application AHB frequency. Higher TRDT values allow stretching the USB response time to IN tokens in order to compensate for longer AHB read access latency to the data FIFO.

*Note: Only accessible in device mode.*

Bit 9 **HNPCAP**: HNP-capable
The application uses this bit to control the OTG_HS controller’s HNP capabilities.
- 0: HNP capability is not enabled.
- 1: HNP capability is enabled.

*Note: Accessible in both device and host modes.*

Bit 8 **SRPCAP**: SRP-capable
The application uses this bit to control the OTG_HS controller’s SRP capabilities. If the core operates as a non-SRP-capable B-device, it cannot request the connected A-device (host) to activate VBUSB and start a session.
- 0: SRP capability is not enabled.
- 1: SRP capability is enabled.

*Note: Accessible in both device and host modes.*

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PHYSEL**: Full speed serial transceiver select
- 0: USB 2.0 external ULPI high-speed PHY.
- 1: USB 1.1 full-speed transceiver.
Bit 5  Reserved, must be kept at reset value.
Bit 4  Reserved, must be kept at reset value.
Bit 3  Reserved, must be kept at reset value.

Bits 2:0  TOCAL[2:0]: FS timeout calibration
The number of PHY clocks that the application programs in this field is added to the full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the line state condition can vary from one PHY to another.
The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock is 0.25 bit times.

<table>
<thead>
<tr>
<th>AHB frequency range (MHz)</th>
<th>TRDT minimum value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>14.2</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>17.2</td>
</tr>
<tr>
<td>17.2</td>
<td>18.5</td>
</tr>
<tr>
<td>18.5</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>21.8</td>
</tr>
<tr>
<td>21.8</td>
<td>24</td>
</tr>
<tr>
<td>24</td>
<td>27.5</td>
</tr>
<tr>
<td>27.5</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 500. TRDT values (HS)

<table>
<thead>
<tr>
<th>AHB frequency range (MHz)</th>
<th>TRDT minimum value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>30</td>
<td>-</td>
</tr>
</tbody>
</table>
### 62.14.5 OTG reset register (OTG_GRSTCTL)

Address offset: 0x10
Reset value: 0x8000 0000

The application uses this register to reset various hardware features inside the core.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>AHBIDL: AHB master idle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicates that the AHB master state machine is in the Idle condition.</td>
</tr>
<tr>
<td>Note:</td>
<td>Accessible in both device and host modes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>DMAREQ: DMA request signal enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit indicates that the DMA request is in progress. Used for debug.</td>
</tr>
<tr>
<td>Bits 29:11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 10:6</th>
<th>TXFNUM[4:0]: Tx FIFO number</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the FIFO number that must be flushed using the Tx FIFO Flush bit. This field must not be changed until the core clears the Tx FIFO Flush bit.</td>
<td></td>
</tr>
<tr>
<td>00000:</td>
<td>Non-periodic Tx FIFO flush in host mode</td>
</tr>
<tr>
<td></td>
<td>Tx FIFO 0 flush in device mode</td>
</tr>
<tr>
<td>00001:</td>
<td>Periodic Tx FIFO flush in host mode</td>
</tr>
<tr>
<td></td>
<td>Tx FIFO 1 flush in device mode</td>
</tr>
<tr>
<td>00010:</td>
<td>Tx FIFO 2 flush in device mode</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>01111:</td>
<td>Tx FIFO 15 flush in device mode</td>
</tr>
<tr>
<td>10000:</td>
<td>Flush all the transmit FIFOs in device or host mode.</td>
</tr>
<tr>
<td>Note:</td>
<td>Accessible in both device and host modes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 5</th>
<th>TXFFLSH: Tx FIFO flush</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction.</td>
<td></td>
</tr>
<tr>
<td>The application must write this bit only after checking that the core is neither writing to the Tx FIFO nor reading from the Tx FIFO. Verify using these registers:</td>
<td></td>
</tr>
<tr>
<td>Read—NAK Effective interrupt ensures the core is not reading from the FIFO</td>
<td></td>
</tr>
<tr>
<td>Write—AHBIDL bit in OTG_GRSTCTL ensures the core is not writing anything to the FIFO.</td>
<td></td>
</tr>
<tr>
<td>Flushing is normally recommended when FIFOs are reconfigured. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk.</td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td>Accessible in both device and host modes.</td>
</tr>
</tbody>
</table>
Bit 4 RXFFLSH: Rx FIFO flush
   The application can flush the entire Rx FIFO using this bit, but must first ensure that the core
   is not in the middle of a transaction.
   The application must only write to this bit after checking that the core is neither reading from
   the Rx FIFO nor writing to the Rx FIFO.
   The application must wait until the bit is cleared before performing any other operations. This
   bit requires 8 clocks (slowest of PHY or AHB clock) to clear.
   Note: Accessible in both device and host modes.

Bit 3 Reserved, must be kept at reset value.

Bit 2 FCRST: Host frame counter reset
   The application writes this bit to reset the (micro-)frame number counter inside the core.
   When the (micro-)frame counter is reset, the subsequent SOF sent out by the core has a
   frame number of 0.
   When application writes "1" to the bit, it might not be able to read back the value as it will get
   cleared by the core in a few clock cycles.
   Note: Only accessible in host mode.

Bit 1 PSRST: Partial soft reset
   Resets the internal state machines but keeps the enumeration info. Could be used to recover
   some specific PHY errors.
   Note: Accessible in both device and host modes.

Bit 0 CSRST: Core soft reset
   Resets the HCLK and PHY clock domains as follows:
   Clears the interrupts and all the CSR register bits except for the following bits:
   – GATEHCLK bit in OTG_PCGCCTL
   – STPPCLK bit in OTG_PCGCCTL
   – FSLSPCS bits in OTG_HCFG
   – DSPD bit in OTG_DCFG
   – SDIS bit in OTG_DCTL
   – OTG_GCCFG register
   All module state machines (except for the AHB slave unit) are reset to the Idle state, and all
   the transmit FIFOs and the receive FIFO are flushed.
   Any transactions on the AHB Master are terminated as soon as possible, after completing the
   last data phase of an AHB transfer. Any transactions on the USB are terminated immediately.
   The application can write to this bit any time it wants to reset the core. This is a self-clearing
   bit and the core clears this bit after all the necessary logic is reset in the core, which can take
   several clocks, depending on the current state of the core. Once this bit has been cleared,
   the software must wait at least 3 PHY clocks before accessing the PHY domain
   (synchronization delay). The software must also check that bit 31 in this register is set to 1
   (AHB Master is Idle) before starting any operation.
   Typically, the software reset is used during software development and also when the user
   dynamically changes the PHY selection bits in the above listed USB configuration registers.
   When you change the PHY, the corresponding clock for the PHY is selected and used in the
   PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper
   operation.
   Note: Accessible in both device and host modes.
62.14.6 OTG core interrupt register (OTG_GINTSTS)

Address offset: 0x014
Reset value: 0x0400 0020

This register interrupts the application for system-level events in the current mode (device mode or host mode).

Some of the bits in this register are valid only in host mode, while others are valid in device mode only. This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>WKUPINT: Resume/remote wakeup detected interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Wakeup interrupt during suspend(L2) or LPM(L1) state.</td>
</tr>
<tr>
<td></td>
<td>- During suspend(L2):</td>
</tr>
<tr>
<td></td>
<td>In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wakeup is detected on the USB.</td>
</tr>
<tr>
<td></td>
<td>- During LPM(L1):</td>
</tr>
<tr>
<td></td>
<td>This interrupt is asserted for either host initiated resume or device initiated remote wakeup on USB.</td>
</tr>
<tr>
<td>Note:</td>
<td>Accessible in both device and host modes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>SRQINT: Session request/new session detected interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when VBUS is in the valid range for a B-peripheral device. Accessible in both device and host modes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>DISCINT: Disconnect detected interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Asserted when a device disconnect is detected.</td>
</tr>
<tr>
<td>Note:</td>
<td>Only accessible in host mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>CIDSCHG: Connector ID status change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The core sets this bit when there is a change in connector ID status.</td>
</tr>
<tr>
<td>Note:</td>
<td>Accessible in both device and host modes.</td>
</tr>
</tbody>
</table>
Bit 27 **LPMINT**: LPM interrupt
In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.
In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG_GLPMCFG).
This field is valid only if the LPMEN bit in OTG_GLPMCFG is set to 1.

Bit 26 **PTXFE**: Periodic Tx FIFO empty
Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).
*Note: Only accessible in host mode.*

Bit 25 **HCINT**: Host channels interrupt
The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.
*Note: Only accessible in host mode.*

Bit 24 **HPRTINT**: Host port interrupt
The core sets this bit to indicate a change in port status of one of the OTG_HS controller ports in host mode. The application must read the OTG_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPRT register to clear this bit.
*Note: Only accessible in host mode.*

Bit 23 **RSTDET**: Reset detected interrupt
In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.
*Note: Only accessible in device mode.*

Bit 22 **DATAFSUSP**: Data fetch suspended
This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or request queue space. This interrupt is used by the application for an endpoint mismatch algorithm. For example, after detecting an endpoint mismatch, the application:
- Sets a global nonperiodic IN NAK handshake
- Disables IN endpoints
- Flushes the FIFO
- Determines the token sequence from the IN token sequence learning queue
- Re-enables the endpoints
Clears the global nonperiodic IN NAK handshake if the global nonperiodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an "IN token received when FIFO empty" interrupt. The OTG then sends a NAK response to the host. To avoid this scenario, the application can check the FetSusp interrupt in OTG_GINTSTS, which ensures that the FIFO is full before clearing a global NAK handshake. Alternatively, the application can mask the "IN token received when FIFO empty" interrupt when clearing a global IN NAK handshake.
Bit 21  **IPXFR:** Incomplete periodic transfer  
In host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending, which are scheduled for the current frame.

**INCOMPISOOUT:** Incomplete isochronous OUT transfer  
In device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Bit 20  **ISOIXFR:** Incomplete isochronous IN transfer  
The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.  
*Note:* Only accessible in device mode.

Bit 19  **OEPINT:** OUT endpoint interrupt  
The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPINTx register to clear this bit.  
*Note:* Only accessible in device mode.

Bit 18  **IEPINT:** IN endpoint interrupt  
The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPINTx register to clear this bit.  
*Note:* Only accessible in device mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15  **EOPF:** End of periodic frame interrupt  
Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.  
*Note:* Only accessible in device mode.

Bit 14  **ISOODRP:** Isochronous OUT packet dropped interrupt  
The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.  
*Note:* Only accessible in device mode.

Bit 13  **ENUMDNE:** Enumeration done  
The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.  
*Note:* Only accessible in device mode.

Bit 12  **USBRST:** USB reset  
The core sets this bit to indicate that a reset is detected on the USB.  
*Note:* Only accessible in device mode.
Bit 11 **USBSUSP**: USB suspend

The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.

*Note: Only accessible in device mode.*

Bit 10 **ESUSP**: Early suspend

The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

*Note: Only accessible in device mode.*

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFF**: Global OUT NAK effective

Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).

*Note: Only accessible in device mode.*

Bit 6 **GINAKEFF**: Global IN non-periodic NAK effective

Indicates that the Set global non-periodic IN NAK bit in the OTG_DCTL register (SGINAK bit in OTG_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG_DCTL register (CGINAK bit in OTG_DCTL).

This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

*Note: Only accessible in device mode.*

Bit 5 **NPTXFE**: Non-periodic Tx FIFO empty

This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

*Note: Accessible in host mode only.*

Bit 4 **RXFLVL**: Rx FIFO non-empty

Indicates that there is at least one packet pending to be read from the Rx FIFO.

*Note: Accessible in both host and device modes.*

Bit 3 **SOF**: Start of frame

In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.

In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.

*Note: This register may return ‘1’ if read immediately after power on reset. If the register bit reads ‘1’ immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.*

*Note: Accessible in both host and device modes.*
Bit 2 OTGINT: OTG interrupt
The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_GOTGINT register to clear this bit.

*Note: Accessible in both host and device modes.*

Bit 1 MMIS: Mode mismatch interrupt
The core sets this bit when the application is trying to access:
– A host mode register, when the core is operating in device mode
– A device mode register, when the core is operating in host mode

The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.

*Note: Accessible in both host and device modes.*

Bit 0 CMOD: Current mode of operation
Indicates the current mode.
0: Device mode
1: Host mode

*Note: Accessible in both host and device modes.*

### 62.14.7 OTG interrupt mask register (OTG_GINTMSK)

Address offset: 0x018
Reset value: 0x0000 0000

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>WUIM</td>
<td>SRQIM</td>
<td>DISCIN</td>
<td>HGM</td>
<td>LPMIM</td>
<td>TM</td>
<td>PTFXM</td>
<td>HCIM</td>
<td>PRTIM</td>
<td>RSTDE</td>
<td>TM</td>
<td>FSUS</td>
<td>IPXFR</td>
<td>MIISO</td>
<td>OXFR</td>
<td>IEPINT</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>EOPF</td>
<td>ISOOD</td>
<td>ENUM</td>
<td>DNEM</td>
<td>USBS</td>
<td>SPMT</td>
<td>ESUSPM</td>
<td>Res</td>
<td>Res</td>
<td>GONA</td>
<td>KEEFM</td>
<td>GINAK</td>
<td>EFFM</td>
<td>NPTXBF</td>
<td>RXFLVM</td>
<td>MMISM</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

2574/2974 RM0455 Rev 3
Bit 31  **WUIM**: Resume/remote wakeup detected interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Accessible in both host and device modes.*

Bit 30  **SRQIM**: Session request/new session detected interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Accessible in both host and device modes.*

Bit 29  **DISCINT**: Disconnect detected interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in host mode.*

Bit 28  **CIDSCHGM**: Connector ID status change mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Accessible in both host and device modes.*

Bit 27  **LPMINTM**: LPM interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Accessible in both host and device modes.*

Bit 26  **PTXFEM**: Periodic Tx FIFO empty mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in host mode.*

Bit 25  **HCIM**: Host channels interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in host mode.*

Bit 24  **PRTIM**: Host port interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in host mode.*

Bit 23  **RSTDETM**: Reset detected interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt  
*Note: Only accessible in device mode.*

Bit 22  **FSUSPM**: Data fetch suspended mask  
0: Masked interrupt  
1: Unmasked interrupt  
Only accessible in peripheral mode.
Bit 21 **IPXFRM**: Incomplete periodic transfer mask
- 0: Masked interrupt
- 1: Unmasked interrupt
*Note:* *Only accessible in host mode.*

**IISOXXFRM**: Incomplete isochronous OUT transfer mask
- 0: Masked interrupt
- 1: Unmasked interrupt
*Note:* *Only accessible in device mode.*

Bit 20 **IISOIXFRM**: Incomplete isochronous IN transfer mask
- 0: Masked interrupt
- 1: Unmasked interrupt
*Note:* *Only accessible in device mode.*

Bit 19 **OEPINT**: OUT endpoints interrupt mask
- 0: Masked interrupt
- 1: Unmasked interrupt
*Note:* *Only accessible in device mode.*

Bit 18 **IEPINT**: IN endpoints interrupt mask
- 0: Masked interrupt
- 1: Unmasked interrupt
*Note:* *Only accessible in device mode.*

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPFM**: End of periodic frame interrupt mask
- 0: Masked interrupt
- 1: Unmasked interrupt
*Note:* *Only accessible in device mode.*

Bit 14 **ISOODRPM**: Isochronous OUT packet dropped interrupt mask
- 0: Masked interrupt
- 1: Unmasked interrupt
*Note:* *Only accessible in device mode.*

Bit 13 **ENUMDNEM**: Enumeration done mask
- 0: Masked interrupt
- 1: Unmasked interrupt
*Note:* *Only accessible in device mode.*

Bit 12 **USBRST**: USB reset mask
- 0: Masked interrupt
- 1: Unmasked interrupt
*Note:* *Only accessible in device mode.*

Bit 11 **USBSUSPM**: USB suspend mask
- 0: Masked interrupt
- 1: Unmasked interrupt
*Note:* *Only accessible in device mode.*

Bit 10 **ESUSPM**: Early suspend mask
- 0: Masked interrupt
- 1: Unmasked interrupt
*Note:* *Only accessible in device mode.*
Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFFM**: Global OUT NAK effective mask
   0: Masked interrupt
   1: Unmasked interrupt
   **Note**: Only accessible in device mode.

Bit 6 **GINAKEFFM**: Global non-periodic IN NAK effective mask
   0: Masked interrupt
   1: Unmasked interrupt
   **Note**: Only accessible in device mode.

Bit 5 **NPTXFEM**: Non-periodic Tx FIFO empty mask
   0: Masked interrupt
   1: Unmasked interrupt
   **Note**: Only accessible in host mode.

Bit 4 **RXFLVLM**: Receive FIFO non-empty mask
   0: Masked interrupt
   1: Unmasked interrupt
   **Note**: Accessible in both device and host modes.

Bit 3 **SOFM**: Start of frame mask
   0: Masked interrupt
   1: Unmasked interrupt
   **Note**: Accessible in both device and host modes.

Bit 2 **OTGINT**: OTG interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt
   **Note**: Accessible in both device and host modes.

Bit 1 **MMISM**: Mode mismatch interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt
   **Note**: Accessible in both device and host modes.

Bit 0 Reserved, must be kept at reset value.
62.14.8  OTG receive status debug read register (OTG_GRXSTSR)

Address offset for read: 0x01C
Reset value: 0x0000 0000

This description is for register OTG_GRXSTSR in Device mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:28  Reserved, must be kept at reset value.

Bit 27  **STSPHST**: Status phase start
Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25  Reserved, must be kept at reset value.

Bits 24:21  **FRMNUM[3:0]**: Frame number
This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17  **PKTSTS[3:0]**: Packet status
Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15  **DPID[1:0]**: Data PID
Indicates the data PID of the received OUT data packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4  **BCNT[10:0]**: Byte count
Indicates the byte count of the received data packet.

Bits 3:0  **EPNUM[3:0]**: Endpoint number
Indicates the endpoint number to which the current received packet belongs.
62.14.9 OTG receive status debug read [alternate] (OTG_GRXSTSR)

Address offset for read: 0x01C
Reset value: 0x0000 0000

This description is for register OTG_GRXSTSR in Host mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>PKTSTS[3:0]</th>
<th></th>
<th></th>
<th></th>
<th>DPID</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS[3:0]:** Packet status
  - Indicates the status of the received packet
  - 0010: IN data packet received
  - 0011: IN transfer completed (triggers an interrupt)
  - 0101: Data toggle error (triggers an interrupt)
  - 0111: Channel halted (triggers an interrupt)
  - Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
  - Indicates the data PID of the received packet
  - 00: DATA0
  - 10: DATA1
  - 01: DATA2
  - 11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count
  - Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number
  - Indicates the channel number to which the current received packet belongs.
### 62.14.10 OTG status read and pop registers (OTG_GRXSTSP)

Address offset for pop: 0x020  
Reset value: 0x0000 0000  

This description is for register OTG_GRXSTSP in Device mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
| 27   | **STSPHST**: Status phase start  
Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern. |
| 26:25 | Reserved, must be kept at reset value. |
| 24:21 | **FRMNUM[3:0]**: Frame number  
This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported. |
| 20:17 | **PKTSTS[3:0]**: Packet status  
Indicates the status of the received packet  
0001: Global OUT NAK (triggers an interrupt)  
0010: OUT data packet received  
0011: OUT transfer completed (triggers an interrupt)  
0100: SETUP transaction completed (triggers an interrupt)  
0110: SETUP data packet received  
Others: Reserved |
| 16:12 |  
| 11:0  |  
| 15:10 |  
| 9:8   |  
| 7:6   |  
| 5:4   |  
| 3:2   |  
| 1:0   |  

#### Bit Description

- **STSPHST**: Status phase start  
- **FRMNUM[3:0]**: Frame number  
- **PKTSTS[3:0]**: Packet status
62.14.11  OTG status read and pop registers [alternate] (OTG_GRXSTSP)

Address offset for pop: 0x020
Reset value: 0x0000 0000

This description is for register OTG_GRXSTSP in Host mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:21  Reserved, must be kept at reset value.

Bits 20:17  PKTSTS[3:0]: Packet status
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved
Bits 16:15 **DPID[1:0]**: Data PID
- Indicates the data PID of the received packet
- 00: DATA0
- 10: DATA1
- 01: DATA2
- 11: MDATA

Bits 14:4 **BCNT[10:0]**: Byte count
- Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]**: Channel number
- Indicates the channel number to which the current received packet belongs.

### 62.14.12 OTG receive FIFO size register (OTG_GRXFSIZ)

**Address offset:** 0x024
**Reset value:** 0x0000 0400

The application can program the RAM size that must be allocated to the Rx FIFO.

![FIFO size register diagram]

- **RXFD[15:0]**: Rx FIFO depth
  - This value is in terms of 32-bit words.
  - Maximum value is 1024
  - Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

### 62.14.13 OTG host non-periodic transmit FIFO size register (OTG_HNPTXFSIZ)/Endpoint 0 Transmit FIFO size (OTG_DIEPTXF0)

**Address offset:** 0x028
**Reset value:** 0x0200 0200

- **NPTXFD/TX0FD[15:0]**
- **NPTXFSA/TX0FSA[15:0]**
Host mode

Bits 31:16 **NPTXFD[15:0]**: Non-periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **NPTXFSA[15:0]**: Non-periodic transmit RAM start address
This field configures the memory start address for non-periodic transmit FIFO RAM.

Device mode

Bits 31:16 **TX0FD**: Endpoint 0 Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **TX0FSA**: Endpoint 0 transmit RAM start address
This field configures the memory start address for the endpoint 0 transmit FIFO RAM.

**62.14.14 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)**

Address offset: 0x02C
Reset value: 0x0008 0400

**Note:** *In device mode, this register is not valid.*

This read-only register contains the free space information for the non-periodic Tx FIFO and the non-periodic transmit request queue.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>NPTXQTOP[6:0]</td>
<td>NPTQXSAV[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Res.</td>
<td>NPTXQTOP[6:0]</td>
<td>NPTQXSAV[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPTXFSA[15:0]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>
Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **NPTXQTOP[6:0]**: Top of the non-periodic transmit request queue
Entry in the non-periodic Tx request queue that is currently being processed by the MAC.
Bits 30:27: Channel/endpoint number
Bits 26:25:
00: IN/OUT token
01: Zero-length transmit packet (device IN/host OUT)
11: Channel halt command
Bit 24: Terminate (last entry for selected channel/endpoint)

Bits 23:16 **NPTQXSAV[7:0]**: Non-periodic transmit request queue space available
Indicates the amount of free space available in the non-periodic transmit request queue.
This queue holds both IN and OUT requests.
0: Non-periodic transmit request queue is full
1: 1 location available
2: locations available
n: n locations available (0 ≤ n ≤ 8)
Others: Reserved

Bits 15:0 **NPTXFSAV[15:0]**: Non-periodic Tx FIFO space available
Indicates the amount of free space available in the non-periodic Tx FIFO.
Values are in terms of 32-bit words.
0: Non-periodic Tx FIFO is full
1: 1 word available
2: 2 words available
n: n words available (where 0 ≤ n ≤ 512)
Others: Reserved

### 62.14.15 OTG general core configuration register (OTG_GCCFG)

Address offset: 0x038
Reset value: 0x0000 XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
Bits 31:22  Reserved, must be kept at reset value.

Bit 21 **VBDEN**: USB VBUS detection enable  
Enables VBUS sensing comparators to detect VBUS valid levels on the VBUS PAD for USB host and device operation. If HNP and/or SRP support is enabled, VBUS comparators are automatically enabled independently of VBDEN value.  
0 = VBUS detection disabled  
1 = VBUS detection enabled

Bit 20 **SDEN**: Secondary detection (SD) mode enable  
This bit is set by the software to put the BCD into SD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly

Bit 19 **PDEN**: Primary detection (PD) mode enable  
This bit is set by the software to put the BCD into PD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly.

Bit 18 **DCDEN**: Data contact detection (DCD) mode enable  
This bit is set by the software to put the BCD into DCD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly.

Bit 17 **BCDEN**: Battery charging detector (BCD) enable  
This bit is set by the software to enable the BCD support within the USB device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD should be placed in OFF mode by clearing this bit to ‘0’ in order to allow the normal USB operation.

Bit 16 **PWRDWN**: Power down control  
Used to activate the transceiver in transmission/reception. When reset, the transceiver is kept in power-down. When set, the BCD function must be off (BCDEN=0).  
0 = USB FS transceiver disabled  
1 = USB FS transceiver enabled

Bits 15:4  Reserved, must be kept at reset value.

Bit 3 **PS2DET**: DM pull-up detection status  
This bit is active only during PD and gives the result of comparison between DM voltage level and VLGC threshold. In normal situation, the DM level should be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.  
0: Normal port detected (connected to SDP, CDP or DCP)  
1: PS2 port or proprietary charger detected

Bit 2 **SDET**: Secondary detection (SD) status  
This bit gives the result of SD.  
0: CDP detected  
1: DCP detected

Bit 1 **PDET**: Primary detection (PD) status  
This bit gives the result of PD.  
0: no BCD support detected (connected to SDP or proprietary device).  
1: BCD support detected (connected to CDP or DCP).

Bit 0 **DCDET**: Data contact detection (DCD) status  
This bit gives the result of DCD.  
0: data lines contact not detected  
1: data lines contact detected
62.14.16 OTG core ID register (OTG_CID)

Address offset: 0x03C
Reset value: 0x0000 2300

This is a register containing the Product ID as reset value.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 PRODUCT_ID[31:0]: Product ID field
Application-programmable ID field.

62.14.17 OTG core LPM configuration register (OTG_GLPMCFG)

Address offset: 0x54
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>-------</td>
<td>-------------</td>
<td>------</td>
<td>----------</td>
<td>-------------</td>
<td>------</td>
<td>----</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 ENBESL: Enable best effort service latency
This bit enables the BESL feature as defined in the LPM errata:
0: The core works as described in the following document:
USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
1: The core works as described in the LPM Errata:
Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
Note: Only the updated behavior (described in LPM Errata) is considered in this document and so the ENBESL bit should be set to ‘1’ by application SW.

Bits 27:25 LPMRCNTSTS[2:0]: LPM retry count status
Number of LPM host retries still remaining to be transmitted for the current LPM sequence.
Note: Accessible only in host mode.
Bit 24 **SNDLPM**: Send LPM transaction
When the application software sets this bit, an LPM transaction containing two tokens, EXT and LPM is sent. The hardware clears this bit once a valid response (STALL, NYET, or ACK) is received from the device or the core has finished transmitting the programmed number of LPM retries.

*Note: This bit must be set only when the host is connected to a local port.*
*Note: Accessible only in host mode.*

Bits 23:21 **LPMRCNT[2:0]**: LPM retry count
When the device gives an ERROR response, this is the number of additional LPM retries that the host performs until a valid device response (STALL, NYET, or ACK) is received.

*Note: Accessible only in host mode.*

Bits 20:17 **LPMCHIDX[3:0]**: LPM Channel Index
The channel number on which the LPM transaction has to be applied while sending an LPM transaction to the local device. Based on the LPM channel index, the core automatically inserts the device address and endpoint number programmed in the corresponding channel into the LPM transaction.

*Note: Accessible only in host mode.*

Bit 16 **L1RSMOK**: Sleep state resume OK
Indicates that the device or host can start resume from Sleep state. This bit is valid in LPM sleep (L1) state. It is set in sleep mode after a delay of 50 μs (T_{L1Residency}).

This bit is reset when SLPSTS is 0.

1: The application or host can start resume from Sleep state
0: The application or host cannot start resume from Sleep state

Bit 15 **SLPSTS**: Port sleep status

**Device mode:**
This bit is set as long as a Sleep condition is present on the USB bus. The core enters the Sleep state when an ACK response is sent to an LPM transaction and the T_{L1TokenRetry} timer has expired. To stop the PHY clock, the application must set the STPPCLK bit in OTG_PCGCCTL, which asserts the PHY suspend input signal.

The application must rely on SLPSTS and not ACK in LPMRSP to confirm transition into sleep.

The core comes out of sleep:
– When there is any activity on the USB linestate
– When the application writes to the RWUSIG bit in OTG_DCTL or when the application resets or soft-disconnects the device.

**Host mode:**
The host transitions to Sleep (L1) state as a side-effect of a successful LPM transaction by the core to the local port with ACK response from the device. The read value of this bit reflects the current Sleep status of the port.

The core clears this bit after:
– The core detects a remote L1 wakeup signal,
– The application sets the PRST bit or the PRES bit in the OTG_HPRT register, or
– The application sets the L1Resume/ remote wakeup detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT bit in OTG_GINTSTS, respectively).

0: Core not in L1
1: Core in L1
Bits 14:13  **LPMRSP[1:0]:** LPM response  
**Device mode:**  
The response of the core to LPM transaction received is reflected in these two bits.  
**Host mode:**  
Handshake response received from local device for LPM transaction  
11: ACK  
10: NYET  
01: STALL  
00: ERROR (No handshake response)

Bit 12  **L1DSEN:** L1 deep sleep enable  
Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit should be set to ‘1’ by application SW in all the cases.

Bits 11:8  **BESLTHRS[3:0]:** BESL threshold  
**Device mode:**  
The core puts the PHY into deep low power mode in L1 when BESL value is greater than or equal to the value defined in this field BESL_Thres[3:0].  
**Host mode:**  
The core puts the PHY into deep low power mode in L1. BESLTHRS[3:0] specifies the time for which resume signaling is to be reflected by host (T_{L1HubDrvResume2}) on the USB bus when it detects device initiated resume.  
BESLTHRS must not be programmed with a value greater than 1100b in host mode, because this exceeds maximum T_{L1HubDrvResume2}.

**Thres[3:0] host mode resume signaling time (μs):**  
0000: 75  
0001: 100  
0010: 150  
0011: 250  
0100: 350  
0101: 450  
0110: 950  
All other values: reserved

Bit 7  **L1SSEN:** L1 Shallow Sleep enable  
Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit should be set to ‘1’ by application SW in all the cases.

Bit 6  **REMWAKE:** bRemoteWake value  
**Host mode:**  
The value of remote wake up to be sent in the windex field of LPM transaction.  
**Device mode (read-only):**  
This field is updated with the received LPM token bRemoteWake bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.
Bits 5:2 **BESL[3:0]:** Best effort service latency

- **Host mode:**
  The value of BESL to be sent in an LPM transaction. This value is also used to initiate resume for a duration $T_{\text{HubDrvResume}}$ for host initiated resume.

- **Device mode (read-only):**
  This field is updated with the received LPM token BESL bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

**BESL[3:0]TBESL (μs)**

- 0000: 125
- 0001: 150
- 0010: 200
- 0011: 300
- 0100: 400
- 0101: 500
- 0110: 1000
- 0111: 2000
- 1000: 3000
- 1001: 4000
- 1010: 5000
- 1011: 6000
- 1100: 7000
- 1101: 8000
- 1110: 9000
- 1111: 10000

Bit 1 **LPMACK:** LPM token acknowledge enable

- Handshake response to LPM token preprogrammed by device application software.
  - 1: ACK
  - Even though ACK is preprogrammed, the core device responds with ACK only on successful LPM transaction. The LPM transaction is successful if:
    - No PID/CRC5 errors in either EXT token or LPM token (else ERROR)
    - Valid bLinkState = 0001B (L1) received in LPM transaction (else STALL)
    - No data pending in transmit queue (else NYET).
  - 0: NYET

The preprogrammed software bit is over-ridden for response to LPM token when:

- The received bLinkState is not L1 (STALL response), or
- An error is detected in either of the LPM token packets because of corruption (ERROR response).

**Note:** Accessible only in device mode.

Bit 0 **LPMEN:** LPM support enable

- The application uses this bit to control the OTG_HS core LPM capabilities.
- If the core operates as a non-LPM-capable host, it cannot request the connected device or hub to activate LPM mode.
- If the core operates as a non-LPM-capable device, it cannot respond to any LPM transactions.
  - 0: LPM capability is not enabled
  - 1: LPM capability is enabled
62.14.18 OTG host periodic transmit FIFO size register
(OTG_HPTXFSIZ)
Address offset: 0x100
Reset value: 0x0400 0800

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>PTXFSIZ[15:0]: Host periodic Tx FIFO depth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This value is in terms of 32-bit words.</td>
</tr>
<tr>
<td></td>
<td>Minimum value is 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>PTXSA[15:0]: Host periodic Tx FIFO start address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This field configures the memory start address for periodic transmit FIFO RAM.</td>
</tr>
</tbody>
</table>

62.14.19 OTG device IN endpoint transmit FIFO x size register
(OTG_DIEPTXFX)
Address offset: 0x104 + 0x04 * (x - 1), (x = 1 to 8)
Reset value: Block 1: 0x0200 0400
Reset value: Block 2: 0x0200 0600
Reset value: Block 3: 0x0200 0800
Reset value: Block 4: 0x0200 0A00
Reset value: Block 5: 0x0200 0C00
Reset value: Block 6: 0x0200 0E00
Reset value: Block 7: 0x0200 1000
Reset value: Block 8: 0x0200 1200

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>INEPTXFD[15:0]:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Endpoint Tx FIFO depth</td>
</tr>
<tr>
<td></td>
<td>This value is in terms of 32-bit words.</td>
</tr>
<tr>
<td></td>
<td>Minimum value is 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>INEPTXSA[15:0]:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Endpoint Tx FIFO start address</td>
</tr>
<tr>
<td></td>
<td>This field configures the memory start address for endpoint transmit FIFO RAM.</td>
</tr>
</tbody>
</table>
62.14.20  Host-mode registers

Bit values in the register descriptions are expressed in binary unless otherwise specified. Host-mode registers affect the operation of the core in the host mode. Host mode registers must not be accessed in device mode, as the results are undefined. Host mode registers can be categorized as follows:

62.14.21  OTG host configuration register (OTG_HCFG)

Address offset: 0x400
Reset value: 0x0000 0000
This register configures the core after power-on. Do not make changes to this register after initializing the host.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:27  Reserved, must be kept at reset value.

Bit 26  PERSSCHEDENA: Enable periodic scheduling
Applicable in host scatter/gather DMA mode only. Enables periodic scheduling within the core. Initially, the bit is res and the core does not process any periodic channels. As soon as this bit is set, the core gets ready to start scheduling periodic channels and sets OTG_HCFG.PERSCHEDSTAT. The setting of PERSCHEDSTAT indicates the core has enabled periodic scheduling. Once PERSSCHEDENA is set, the application is not supposed to reset the bit unless PERSCHEDSTAT is set. As soon as this bit is reset, the core gets ready to stop scheduling periodic channels and resets HCFG. PerSchedStat. In non-Scatter/Gather DMA mode, this bit is reserved.

Bits 25:24  FRLSTEN[1:0]: Frame list entries
The value in the register specifies the number of entries in the Frame list. This field is valid only in Scatter/Gather DMA mode.
2'b00: Reserved
2'b01: 8 Entries
2'b10: 16 Entries
2'b11: 32 Entries In non-Scatter/Gather
Bit 23 DESCDMA: Enable scatter/gather DMA in host mode
The application can set this bit during initialization to enable the Scatter/Gather DMA operation. This bit must be modified only once after a reset. The following combinations are available for programming:
- OTG_GAHBCFG.DMAEN=0,OTG_HCFG.DESCDMA=0 => Slave mode
- OTG_GAHBCFG.DMAEN=0,OTG_HCFG.DESCDMA=1 => Invalid
- OTG_GAHBCFG.DMAEN=1,OTG_HCFG.DESCDMA=0 => Buffered DMA mode
- OTG_GAHBCFG.DMAEN=1,OTG_HCFG.DESCDMA=1 => Scatter/Gather DMA mode

Bits 22:3 Reserved, must be kept at reset value.

Bit 2 FSLSS: FS- and LS-only support
The application uses this bit to control the core’s enumeration speed. Using this bit, the application can make the core enumerate as an FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming.

Bits 1:0 FSLSPCS[1:0]: FS/LS PHY clock select
- When the core is in FS host mode:
  - 01: PHY clock is running at 48 MHz
  - Others: Reserved
- When the core is in LS host mode:
  - 00: Reserved
  - 01: Select 48 MHz PHY clock frequency
  - 10: Select 6 MHz PHY clock frequency
  - 11: Reserved

Note: The FSLSPCS must be set on a connection event according to the speed of the connected device (after changing this bit, a software reset must be performed).

62.14.22 OTG host frame interval register (OTG_HFIR)
Address offset: 0x404
Reset value: 0x0000 EA60
This register stores the frame interval information for the current speed to which the OTG_HS controller has enumerated.
Bits 31:17  Reserved, must be kept at reset value.

Bit 16  **RLDCTRL**: Reload control

This bit allows dynamic reloading of the HFIR register during run time.

- **0**: The HFIR cannot be reloaded dynamically
- **1**: The HFIR can be dynamically reloaded during run time.

This bit needs to be programmed during initial configuration and its value must not be changed during run time.

**Caution**: RLDCTRL = 0 is not recommended.

Bits 15:0  **FRIVL[15:0]**: Frame interval

The value that the application programs to this field, specifies the interval between two consecutive micro-SOFs (HS) or Keep-Alive tokens (LS). This field contains the number of PHY clocks that constitute the required frame interval. The application can write a value to this register only after the port enable bit of the host port control and status register (PENA bit in OTG_HPRT) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY clock select field of the host configuration register (FSLSPCS in OTG_HCFG). Do not change the value of this field after the initial configuration, unless the RLDCTRL bit is set. In such case, the FRIVL is reloaded with each SOF event.

- Frame interval = 125 μs × (FRIVL - 1) in high speed operation (PHYSEL = 0)
- Frame interval = 1 ms × (FRIVL - 1) in low/full speed operation (PHYSEL = 1)

### 62.14.23  OTG host frame number/frame time remaining register (OTG_HFNUM)

Address offset: 0x408

Reset value: 0x0000 3FFF

This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current frame.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FTREM[15:0]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FRNUM[15:0]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16  **FTREM[15:0]**: Frame time remaining

Indicates the amount of time remaining in the current frame, in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame interval register and a new SOF is transmitted on the USB.

Bits 15:0  **FRNUM[15:0]**: Frame number

This field increments when a new SOF is transmitted on the USB, and is cleared to 0 when it reaches 0x3FFF.
62.14.24 OTG_Host periodic transmit FIFO/queue status register
(OTG_HPTXSTS)

Address offset: 0x410
Reset value: 0x0008 0100

This read-only register contains the free space information for the periodic Tx FIFO and the periodic transmit request queue.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTXQTOP[7:0]</td>
<td>PTXQSAV[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTXFSAVL[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 PTXQTOP[7:0]: Top of the periodic transmit request queue
This indicates the entry in the periodic Tx request queue that is currently being processed by the MAC.
This register is used for debugging.
Bit 31: Odd/Even frame
0: send in even frame
1: send in odd frame
Bits 30:27: Channel/endpoint number
Bits 26:25: Type
00: IN/OUT
01: Zero-length packet
11: Disable channel command
Bit 24: Terminate (last entry for the selected channel/endpoint)

Bits 23:16 PTXQSAV[7:0]: Periodic transmit request queue space available
Indicates the number of free locations available to be written in the periodic transmit request queue. This queue holds both IN and OUT requests.
00: Periodic transmit request queue is full
01: 1 location available
10: 2 locations available
bxn: n locations available (0 ≤ n ≤ 8)
Others: Reserved

Bits 15:0 PTXFSAVL[15:0]: Periodic transmit data FIFO space available
Indicates the number of free locations available to be written to in the periodic Tx FIFO.
Values are in terms of 32-bit words
0000: Periodic Tx FIFO is full
0001: 1 word available
0010: 2 words available
bxn: n words available (where 0 ≤ n ≤ PTXFD)
Others: Reserved
62.14.25 OTG host all channels interrupt register (OTG_HAINT)

Address offset: 0x414
Reset value: 0x0000 0000

When a significant event occurs on a channel, the host all channels interrupt register
interrupts the application using the host channels interrupt bit of the core interrupt register
(HCINT bit in OTG_GINTSTS). This is shown in Figure 720. There is one interrupt bit per
channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the
application sets and clears bits in the corresponding host channel-x interrupt register.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

HAINT[15:0]
```

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 HAIN[15:0]: Channel interrupts
One bit per channel: Bit 0 for Channel 0, bit 15 for Channel 15

62.14.26 OTG host all channels interrupt mask register
(OTG_HAINTMSK)

Address offset: 0x418
Reset value: 0x0000 0000

The host all channel interrupt mask register works with the host all channel interrupt register
to interrupt the application when an event occurs on a channel. There is one interrupt mask
bit per channel, up to a maximum of 16 bits.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

HAINTM[15:0]
```

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 HAINTM[15:0]: Channel interrupt mask
0: Masked interrupt
1: Unmasked interrupt
One bit per channel: Bit 0 for channel 0, bit 15 for channel 15
62.14.27 OTG host frame list base address register (OTG_HFLBADDR)

Address offset: 0x41C
Reset value: 0x0000 0000

This register holds the starting address of the frame list information (scatter/gather mode).

<table>
<thead>
<tr>
<th>Bit 31:0</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HFLBADDR[31:16]</td>
<td></td>
<td>Starting address of the frame list (scatter/gather mode).</td>
</tr>
<tr>
<td>HFLBADDR[15:0]</td>
<td></td>
<td>Bits 31:0: HFLBADDR[31:0]: The starting address of the frame list (scatter/gather mode).</td>
</tr>
</tbody>
</table>

62.14.28 OTG host port control and status register (OTG_HPRT)

Address offset: 0x440
Reset value: 0x0000 0000

This register is available only in host mode. Currently, the OTG host supports only one port.

A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port. It is shown in Figure 720. The rc_w1 bits in this register can trigger an interrupt to the application through the host port interrupt bit of the core interrupt register (HPRTINT bit in OTG_GINTSTS). On a port interrupt, the application must read this register and clear the bit that caused the interrupt. For the rc_w1 bits, the application must write a 1 to the bit to clear the interrupt.

<table>
<thead>
<tr>
<th>Bit 31:0</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTCTL[2:0]</td>
<td></td>
<td>Controls the USB port.</td>
</tr>
<tr>
<td>PPWR</td>
<td></td>
<td>Power control.</td>
</tr>
<tr>
<td>PLSTS[1:0]</td>
<td></td>
<td>Port status.</td>
</tr>
<tr>
<td>RES</td>
<td></td>
<td>Reserved.</td>
</tr>
<tr>
<td>PRST</td>
<td></td>
<td>Port reset.</td>
</tr>
<tr>
<td>PSUSP</td>
<td></td>
<td>Port suspend.</td>
</tr>
<tr>
<td>PRES</td>
<td></td>
<td>Port resume.</td>
</tr>
<tr>
<td>POC</td>
<td></td>
<td>Port connect.</td>
</tr>
<tr>
<td>CHNG</td>
<td></td>
<td>Change detected.</td>
</tr>
<tr>
<td>POCA</td>
<td></td>
<td>Port connect asserted.</td>
</tr>
<tr>
<td>PEN</td>
<td></td>
<td>Port enable asserted.</td>
</tr>
<tr>
<td>PCDET</td>
<td></td>
<td>Port connect detected.</td>
</tr>
<tr>
<td>PCSTS</td>
<td></td>
<td>Port connect status.</td>
</tr>
<tr>
<td>rw</td>
<td></td>
<td>Read/write.</td>
</tr>
<tr>
<td>r</td>
<td></td>
<td>Read-only.</td>
</tr>
</tbody>
</table>
Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **PSPD[1:0]:** Port speed
- Indicates the speed of the device attached to this port.
- 01: Full speed
- 10: Low speed
- 11: Reserved
- 00: High speed

Bits 16:13 **PTCTL[3:0]:** Port test control
- The application writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is signaled on the port.
- 0000: Test mode disabled
- 0001: Test_J mode
- 0010: Test_K mode
- 0011: Test_SE0_NAK mode
- 0100: Test_Packet mode
- 0101: Test_Force_Enable
- Others: Reserved

Bit 12 **PPWR:** Port power
- The application uses this field to control power to this port, and the core clears this bit on an overcurrent condition.
- 0: Power off
- 1: Power on

Bits 11:10 **PLSTS[1:0]:** Port line status
- Indicates the current logic level USB data lines
- Bit 10: Logic level of OTG_DP
- Bit 11: Logic level of OTG_DM

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PRST:** Port reset
- When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete.
- 0: Port not in reset
- 1: Port in reset
- The application must leave this bit set for a minimum duration of at least 10 ms to start a reset on the port. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.
- High speed: 50 ms
- Full speed/Low speed: 10 ms

Bit 7 **PSUSP:** Port suspend
- The application sets this bit to put this port in suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the port clock stop bit, which asserts the suspend input pin of the PHY.
- The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wakeup signal is detected or the application sets the port reset bit or port resume bit in this register or the resume/remote wakeup detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT in OTG_GINTSTS, respectively).
- 0: Port not in suspend mode
- 1: Port in suspend mode
Bit 6  **PRES**: Port resume  
The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit.  
If the core detects a USB remote wakeup sequence, as indicated by the port resume/remote wake-up detected interrupt bit of the core interrupt register (WKUPINT bit in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.  
0: No resume driven  
1: Resume driven  

When LPM is enabled and the core is in L1 state, the behavior of this bit is as follows:  
1. The application sets this bit to drive resume signaling on the port.  
2. The core continues to drive the resume signal until a predetermined time specified in BESLTHRS[3:0] field of OTG_GLPMLCFG register.  
3. If the core detects a USB remote wakeup sequence, as indicated by the port L1Resume/Remote L1 Wakeup detected interrupt bit of the core interrupt register (WKUPINT in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit at the end of resume. This bit can be set or cleared by both the core and the application. This bit is cleared by the core even if there is no device connected to the host.

Bit 5  **POCCHNG**: Port overcurrent change  
The core sets this bit when the status of the port overcurrent active bit (bit 4) in this register changes.

Bit 4  **POCA**: Port overcurrent active  
Indicates the overcurrent condition of the port.  
0: No overcurrent condition  
1: Overcurrent condition

Bit 3  **PENCHNG**: Port enable/disable change  
The core sets this bit when the status of the port enable bit 2 in this register changes.

Bit 2  **PENA**: Port enable  
A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set this bit by a register write. It can only clear it to disable the port. This bit does not trigger any interrupt to the application.  
0: Port disabled  
1: Port enabled

Bit 1  **PCDET**: Port connect detected  
The core sets this bit when a device connection is detected to trigger an interrupt to the application using the host port interrupt bit in the core interrupt register (HPRTINT bit in OTG_GINTSTS). The application must write a 1 to this bit to clear the interrupt.

Bit 0  **PCSTS**: Port connect status  
0: No device is attached to the port  
1: A device is attached to the port
### 62.14.29 OTG host channel x characteristics register (OTG_HCCHARx)

Address offset: \(0x500 + 0x20 \times x\), \((x = 0\) to \(15)\)

Reset value: \(0x0000\ 0000\)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rs</td>
<td>rs</td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EPDIR</th>
<th>EPNUM[3:0]</th>
<th>MPSIZ[10:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bit 31 CHENA**: Channel enable
- When Scatter/Gather mode is enabled:
  - 1'b0: Indicates that the descriptor structure is not yet ready
  - 1'b1: Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor
- When Scatter/Gather mode is disabled: This field is set by the application and cleared by the OTG host.
  - 0: Channel disabled
  - 1: Channel enabled

**Bit 30 CHDIS**: Channel disable
- The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel disabled interrupt before treating the channel as disabled.

**Bit 29** Reserved, must be kept at reset value.

**Bits 28:22 DAD[6:0]**: Device address
- This field selects the specific device serving as the data source or sink.

**Bits 21:20 MCNT[1:0]**: Multicount
- This field indicates to the host the number of transactions that must be executed per frame for this periodic endpoint. For non-periodic transfers, this field is not used
  - 00: Reserved. This field yields undefined results
  - 01: 1 transaction
  - 10: 2 transactions per frame to be issued for this endpoint
  - 11: 3 transactions per frame to be issued for this endpoint
- **Note**: This field must be set to at least 01.

**Bits 19:18 EPTYP[1:0]**: Endpoint type
- Indicates the transfer type selected.
  - 00: Control
  - 01: Isochronous
  - 10: Bulk
  - 11: Interrupt

**Bit 17 LSDEV**: Low-speed device
- This field is set by the application to indicate that this channel is communicating to a low-speed device.

**Bit 16** Reserved, must be kept at reset value.
Bit 15 **EPDIR**: Endpoint direction
 Indicates whether the transaction is IN or OUT.
 0: OUT
 1: IN

Bits 14:11 **EPNUM[3:0]**: Endpoint number
 Indicates the endpoint number on the device serving as the data source or sink.

Bits 10:0 **MPSIZ[10:0]**: Maximum packet size
 Indicates the maximum packet size of the associated endpoint.

### 62.14.30 OTG host channel x split control register (OTG_HCSPLTx)

Address offset: 0x504 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>SPLITEN</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>COMPLSPLT</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31 **SPLITEN**: Split enable
 The application sets this bit to indicate that this channel is enabled to perform split transactions.

Bits 30:17 Reserved, must be kept at reset value.

Bit 16 **COMPLSPLT**: Do complete split
 The application sets this bit to request the OTG host to perform a complete split transaction.

Bits 15:14 **XACTPOS[1:0]**: Transaction position
 This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction.
 11: All. This is the entire data payload of this transaction (which is less than or equal to 188 bytes)
 10: Begin. This is the first data payload of this transaction (which is larger than 188 bytes)
 00: Mid. This is the middle payload of this transaction (which is larger than 188 bytes)
 01: End. This is the last payload of this transaction (which is larger than 188 bytes)

Bits 13:7 **HUBADDR[6:0]**: Hub address
 This field holds the device address of the transaction translator’s hub.

Bits 6:0 **PRTADDR[6:0]**: Port address
 This field is the port number of the recipient transaction translator.
62.14.31 OTG host channel x interrupt register (OTG_HCINTx)

Address offset: 0x508 + 0x20 * x, (x = 0 to 15)
Reset value: 0x0000 0000

This register indicates the status of a channel with respect to USB- and AHB-related events. It is shown in Figure 720. The application must read this register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Res.</th>
<th>Res.</th>
<th>DE</th>
<th>DCLST</th>
<th>ROLL</th>
<th>XC</th>
<th>SXACT</th>
<th>ERR</th>
<th>BNA</th>
<th>DTER</th>
<th>FRM</th>
<th>OR</th>
<th>BBERR</th>
<th>TXERR</th>
<th>NYET</th>
<th>ACK</th>
<th>NAK</th>
<th>STALL</th>
<th>AHB</th>
<th>ERR</th>
<th>CHH</th>
<th>XFRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 DESCFLTROLL: Descriptor rollover interrupt.
This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel descriptor list rolls over. For non Scatter/Gather DMA mode, this bit is reserved.

Bit 12 XCSXACTERR: Excessive transaction error.
This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR is not generated for isochronous channels. For non Scatter/Gather DMA mode, this bit is reserved.

Bit 11 BNA: Buffer not available interrupt.
This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the core to process. BNA interrupt is not generated for isochronous channels. For non Scatter/Gather DMA mode, this bit is reserved.

Bit 10 DTER: Data toggle error. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 9 FRMOR: Frame overrun. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 8 BBERR: Babble error. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 7 TXERR: Transaction error. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
Indicates one of the following errors occurred on the USB.
CRC check failure
Timeout
Bit stuff error
False EOP
Bit 6 **NYET**: Not yet ready response received interrupt. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 5 **ACK**: ACK response received/transmitted interrupt. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 4 **NAK**: NAK response received interrupt. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 3 **STALL**: STALL response received interrupt. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 2 **AHBERR**: AHB error
   This error is generated only in Internal DMA mode when an AHB error occurs during an AHB read/write operation. The application can read the corresponding DMA channel address register to get the error address.

Bit 1 **CHH**: Channel halted.
   In non scatter/gather DMA mode indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application.
   In scatter/gather DMA mode, this indicates that transfer completed due to any of the following:
   – EOL being set in descriptor
   – AHB error
   – Excessive transaction errors
   – In response to disable request by the application
   – Babble
   – Stall

Bit 0 **XFRC**: Transfer completed.
   Transfer completed normally without any errors.

### 62.14.32 OTG host channel x interrupt mask register (OTG_HCINTMSKx)

Address offset: 0x50C + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

This register reflects the mask for each channel status described in the previous section.
Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **DESCRLSTROLLMSK**: Descriptor rollover interrupt mask register.
   This bit is valid only when Scatter/Gather DMA mode is enabled.
   In non Scatter/Gather DMA mode, this bit is reserved.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **BNAMS**: Buffer not available interrupt mask register.
   This bit is valid only when Scatter/Gather DMA mode is enabled.
   In non Scatter/Gather DMA mode, this bit is reserved.

Bit 10 **DTERM**: Data toggle error mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 9 **FRMORM**: Frame overrun mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 8 **BBERM**: Babble error mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 7 **TXERM**: Transaction error mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 6 **NYET**: response received interrupt mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 5 **ACKM**: ACK response received/transmitted interrupt mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 4 **NAKM**: NAK response received interrupt mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 3 **STALLM**: STALL response received interrupt mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
   0: Masked interrupt
   1: Unmasked interrupt
Bit 2 **AHBERRM**: AHB error. In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in OTG_HCINTx.

- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 1 **CHHM**: Channel halted mask

- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 0 **XFRCM**: Transfer completed mask

- 0: Masked interrupt
- 1: Unmasked interrupt

### 62.14.33 OTG host channel x transfer size register (OTG_HCTSZx)

Address offset: \(0x510 + 0x20 \times x\), \((x = 0 \text{ to } 15)\)

Reset value: \(0x0000\ 0000\)

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>DO PING</th>
<th>Bit 30-29</th>
<th>DPID[1:0]</th>
<th>Bit 28-19</th>
<th>PKTcnt[9:0]</th>
<th>Bit 18-0</th>
<th>XFRSZ[18:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XFRSZ[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31 **DOPNG**: Do Ping

This bit is used only for OUT transfers. Setting this field to 1 directs the host to do PING protocol.

*Note: Do not set this bit for IN transfers. If this bit is set for IN transfers, it disables the channel.*

Bits 30:29 **DPID[1:0]**: Data PID

The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.

- 00: DATA0
- 01: DATA2
- 10: DATA1
- 11: SETUP (control) / MDATA (non-control)

Bits 28:19 **PKTcnt[9:0]**: Packet count

This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN).

The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion.

Bits 18:0 **XFRSZ[18:0]**: Transfer size

For an OUT, this field is the number of data bytes the host sends during the transfer. For an IN, this field is the buffer size that the application has reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).
62.14.34 OTG host channel x transfer size register (OTG_HCTSIZSGx)

Address offset: 0x510 + 0x20 * x, (x = 0 to 15)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DO**

- **PND[1:0]**
  - Res.
  - Res.
  - Res.
  - Res.
  - Res.
  - Res.
  - Res.
  - Res.
  - Res.
  - Res.
  - Res.
  - Res.
  - Res.
  - Res.

- ** rw**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**NTD[7:0]**

**SCHED_INFO[7:0]**

Bit 31 **DOPNG**: Do Ping

- This bit is used only for OUT transfers. Setting this field to 1 directs the host to do PING protocol.

  *Note*: Do not set this bit for IN transfers. If this bit is set for IN transfers, it disables the channel.

Bits 30:29 **PID[1:0]**: Pid

- The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.

  00: DATA0
  01: DATA2
  10: DATA1
  11: MDATA (non-control) / SETUP (control)
Bits 28:16  Reserved, must be kept at reset value.

Bits 15:8  \textbf{NTD}[7:0]: Number of transfer descriptors

\textbf{Non isochronous}: this value is in terms of number of descriptors. The maximum number of descriptor that can be present in the list is 64. The values can be from 0 to 63.

- 0: 1 descriptor
- 1: 2 descriptors
- \ldots
- 63: 64 descriptors

Others: Reserved

This field indicates the total number of descriptors present in that list. The core will wrap around after servicing NTD number of descriptors for that list.

\textbf{Isochronous}: this field indicates the number of descriptors present in that list.

The possible values for FS are:

- 1: 2 descriptors
- 3: 4 descriptors
- 7: 8 descriptors
- 15: 16 descriptors
- 31: 32 descriptors
- 63: 64 descriptors

Others: Reserved

The possible values for HS are:

- 7: 8 descriptors
- 15: 16 descriptors
- 31: 32 descriptors
- 63: 64 descriptors
- 127: 128 descriptors
- 255: 256 descriptors

Others: Reserved

Bits 7:0  \textbf{SCHED_INFO}[7:0]: Schedule information

Every bit in this 8 bit register indicates scheduling for that microframe. Bit 0 indicates scheduling for 1st microframe and bit 7 indicates scheduling for 8th microframe in that frame.

A value of 0b11111111 indicates that the corresponding interrupt channel is scheduled to issue a token every microframe in that frame. A value of 0b10101010 indicates that the corresponding interrupt channel is scheduled to issue a token every alternate microframe starting with second microframe. Note that this field is applicable only for periodic (isochronous and interrupt) channels.
### 62.14.35 OTG host channel x DMA address register in buffer DMA [alternate] (OTG_HCDMAx)

Address offset: 0x514 + 0x20 * x, (x = 0 to 15)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>DMAADDR[31:16]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>31  30  29  28  27  26  25  24  23  22  21  20  19  18  17  16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Bits 31:0 DMAADDR[31:0]: DMA address
This field holds the start address in the external memory from which the data for the endpoint must be fetched or to which it must be stored. This register is incremented on every AHB transaction.

### 62.14.36 OTG host channel x DMA address register in scatter/gather DMA [alternate] (OTG_HCDMASGx)

Address offset: 0x514 + 0x20 * x, (x = 0 to 15)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>DMASG[31:16]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>31  30  29  28  27  26  25  24  23  22  21  20  19  18  17  16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:3 DMASG[31:3]: DMA scatter/gather information

The DMASG information is composed of two parts DMASG_ADDR = DMASG[31:N] and DMASG_CTD = DMASG[N-1:3].

Non-isochronous case (N = 9):
The DMASG_ADDR field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value.
The DMASG_CTD field holds a value in terms of number of descriptors. The values can be from 0 (1 descriptor) to 63 (64 descriptors). This field indicates the current descriptor processed in the list. This field is updated both by application and the core. For example, if the application enables the channel after programming CTD = 5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of 8 * 5 = 40 bytes (decimal) to DMAAddr.

Isochronous case (N=4 to 11):
The DMASG_ADDR field holds the address of the 2*(NTD+1) bytes of locations in which the isochronous descriptors are present where N is based on NTD as per the following table:

| HS ISOC; NTD = 7: | N = 6 |
| HS ISOC; NTD = 15: | N = 7 |
| HS ISOC; NTD = 31: | N = 8 |
| HS ISOC; NTD = 63: | N = 9 |
| HS ISOC; NTD = 127: | N = 10 |
| HS ISOC; NTD = 255: | N = 11 |
| FS ISOC; NTD = 1: | N = 4 |
| FS ISOC; NTD = 3: | N = 5 |
| FS ISOC; NTD = 7: | N = 6 |
| FS ISOC; NTD = 15: | N = 7 |
| FS ISOC; NTD = 31: | N = 8 |
| FS ISOC; NTD = 63: | N = 9 |

The DMASG_CTD field is based on the current frame/(micro)frame value. Need to be set to zero by application.

Bits 2:0 Reserved, must be kept at reset value.

62.14.37 OTG host channel-n DMA address buffer register (OTG_HCDMABx)

Address offset: 0x51C + 0x20 * x, (x = 0 to 15)
Reset value: 0x0000 0000 (0x0000 0000)

Bits 31:0 HCDMAB[31:0]: DMA address

This register holds the current buffer address (scatter/gather mode).
**62.14.38 Device-mode registers**

These registers must be programmed every time the core changes to device mode.

**62.14.39 OTG device configuration register (OTG_DCFG)**

Address offset: 0x800

Reset value: 0x0220 0000

This register configures the core in device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

| Bit 31:26 | Reserved, must be kept at reset value. |
| Bit 25:24 | **PERSCHIVL[1:0]**: Periodic schedule interval |
|           | This field specifies the amount of time the Internal DMA engine must allocate for fetching periodic IN endpoint data. Based on the number of periodic endpoints, this value must be specified as 25, 50 or 75% of the (micro) frame. |
|           | – When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data |
|           | – When no periodic endpoint is active, then the internal DMA engine services nonperiodic endpoints, ignoring this field |
|           | – After the specified time within a (micro) frame, the DMA switches to fetching nonperiodic endpoints |
|           | 00: 25% of (micro)frame |
|           | 01: 50% of (micro)frame |
|           | 10: 75% of (micro)frame |
|           | 11: Reserved |

**Note:** Periodic Scheduling Interval (PERSCHIVL) must be programmed for Scatter/Gather DMA mode.

| Bit 23 | **DESCDMA**: Enable scatter/gather DMA in device mode |
|        | The application can set this bit during initialization to enable the Scatter/Gather DMA operation. This bit must be modified only once after a reset. The following combinations are available for programming: |
|        | OTG_GAHBCFG.DMAEN=0,OTG_DCFG.DESCDMA=0 => Slave mode |
|        | OTG_GAHBCFG.DMAEN=0,OTG_DCFG.DESCDMA=1 => Invalid |
|        | OTG_GAHBCFG.DMAEN=1,OTG_DCFG.DESCDMA=0 => Buffer DMA mode |
|        | OTG_GAHBCFG.DMAEN=1,OTG_DCFG.DESCDMA=1 => Scatter/Gather DMA mode |

| Bit 22:16 | Reserved, must be kept at reset value. |
Bit 15 **ERRATIM**: Erratic error interrupt mask
   1: Mask early suspend interrupt on erratic error
   0: Early suspend interrupt is generated on erratic error

Bit 14 **XCVRDLY**: Transceiver delay
   Enables or disables delay in ULPI timing during device chirp.
   0: Disable delay (use default timing)
   1: Enable delay to default timing, necessary for some ULPI PHYs

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 **PFIVL[1:0]**: Periodic frame interval
   Indicates the time within a frame at which the application must be notified using the end of periodic frame interrupt. This can be used to determine if all the isochronous traffic for that frame is complete.
   00: 80% of the frame interval
   01: 85% of the frame interval
   10: 90% of the frame interval
   11: 95% of the frame interval

Bits 10:4 **DAD[6:0]**: Device address
   The application must program this field after every SetAddress control command.
   Bit 3 Reserved, must be kept at reset value.

Bit 2 **NZLSOHX**: Non-zero-length status OUT handshake
   The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's status stage.
   1: Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.
   0: Send the received OUT packet to the application (zero-length or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the device endpoint control register.

Bits 1:0 **DSPD[1:0]**: Device speed
   Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected.
   00: High speed
   01: Full speed using HS
   10: Reserved
   11: Full speed using internal FS PHY
62.14.40 OTG device control register (OTG_DCTL)

Address offset: 0x804
Reset value: 0x0000 0002

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:19: Reserved, must be kept at reset value.

Bit 18 **DSBESLRJCT**: Deep sleep BESL reject
Core rejects LPM request with BESL value greater than BESL threshold programmed. NYET response is sent for LPM tokens with BESL value greater than BESL threshold. By default, the deep sleep BESL reject feature is disabled.

Bit 17 **ENCONTONBNA**: Enable continue on BNA
This bit enables the core to continue on BNA for Bulk OUT and INTR OUT endpoints. With this feature enabled, when a Bulk OUT or INTR OUT endpoint receives a BNA interrupt the core starts processing the descriptor that caused the BNA interrupt after the endpoint re-enables the endpoint.
0: After receiving BNA interrupt, the core disables the endpoint. When the endpoint is re-enabled by the application, the core starts processing from the DEPDM descriptor.
1: After receiving BNA interrupt, the core disables the endpoint. When the endpoint is re-enabled by the application, the core starts processing from the descriptor that received the BNA interrupt. It is a one-time programmable after reset bit like any other OTG_DCTL register bit.

Bits 16:12: Reserved, must be kept at reset value.

Bit 11 **POPRGDNE**: Power-on programming done
The application uses this bit to indicate that register programming is completed after a wakeup from power down mode.

Bit 10 **CGONAK**: Clear global OUT NAK
Writing 1 to this field clears the Global OUT NAK.

Bit 9 **SGONAK**: Set global OUT NAK
Writing 1 to this field sets the Global OUT NAK.
The application uses this bit to send a NAK handshake on all OUT endpoints.
The application must set the this bit only after making sure that the Global OUT NAK effective bit in the core interrupt register (GONAKEFF bit in OTG_GINTSTS) is cleared.

Bit 8 **CGINAK**: Clear global IN NAK
Writing 1 to this field clears the Global IN NAK.

Bit 7 **SGINAK**: Set global IN NAK
Writing 1 to this field sets the Global non-periodic IN NAK.
The application uses this bit to send a NAK handshake on all non-periodic IN endpoints.
The application must set this bit only after making sure that the Global IN NAK effective bit in the core interrupt register (GINAKEFF bit in OTG_GINTSTS) is cleared.
Table 501 contains the minimum duration (according to device state) for which the Soft disconnect (SDIS) bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application add some extra delay to the specified minimum duration.

<table>
<thead>
<tr>
<th>Operating speed</th>
<th>Device state</th>
<th>Minimum duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full speed</td>
<td>Suspended</td>
<td>1 ms + 2.5 µs</td>
</tr>
<tr>
<td>Full speed</td>
<td>Idle</td>
<td>2.5 µs</td>
</tr>
<tr>
<td>Full speed</td>
<td>Not Idle or suspended (Performing transactions)</td>
<td>2.5 µs</td>
</tr>
<tr>
<td>High speed</td>
<td>Not Idle or suspended (Performing transactions)</td>
<td>125 µs</td>
</tr>
</tbody>
</table>
62.14.41 OTG device status register (OTG_DSTS)

Address offset: 0x808
Reset value: 0x0000 0010

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from the device all interrupts (OTG_DAINT) register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **DEVLNSTS[1:0]**: Device line status
- Indicates the current logic level USB data lines.
- Bit [23]: Logic level of D+
- Bit [22]: Logic level of D-

Bits 21:8 **FNSOF[13:0]**: Frame number of the received SOF

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ERR**: Erratic error
- The core sets this bit to report any erratic errors.
- Due to erratic errors, the OTG_HS controller goes into suspended state and an interrupt is generated to the application with Early suspend bit of the OTG_GINTSTS register (ESUSP bit in OTG_GINTSTS). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.

Bits 2:1 **ENUMSPD[1:0]**: Enumerated speed
- Indicates the speed at which the OTG_HS controller has come up after speed detection through a chirp sequence.
- 00: High Speed using HS PHY
- 01: Full Speed using HS PHY
- 11: Full speed using embedded FS PHY
- Others: reserved

Bit 0 **SUSPSTS**: Suspend status
- In device mode, this bit is set as long as a suspend condition is detected on the USB. The core enters the suspended state when there is no activity on the USB data lines for a period of 3 ms. The core comes out of the suspend:
  - When there is an activity on the USB data lines
  - When the application writes to the remote wakeup signaling bit in the OTG_DCTL register (RWUSIG bit in OTG_DCTL).
62.14.42  OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)

Address offset: 0x810
Reset value: 0x0000 0000

This register works with each of the OTG_DIEPINTx registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the OTG_DIEPINTx register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAKM**: NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **BNAM**: BNA interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 8 **TXFURM**: FIFO underrun mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **INEPNEM**: IN endpoint NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **INEPNMM**: IN token received with EP mismatch mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **ITTXFEMSK**: IN token received when Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **TOM**: Timeout condition mask (Non-isochronous endpoints)
0: Masked interrupt
1: Unmasked interrupt
Bit 2 **AHBERRM**: AHB error mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 1 **EPDM**: Endpoint disabled interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 0 **XFRCM**: Transfer completed interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

### 62.14.43 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)

Address offset: 0x814

Reset value: 0x0000 0000

This register works with each of the OTG_DOEPINTx registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the OTG_DOEPINTx register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Mask Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>NYETMSK</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>NAKMSK</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>BERRM</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>Res.</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>Res.</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>Res.</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>Res.</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>Res.</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>Res.</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>Res.</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>Res.</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>Res.</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>Res.</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>Res.</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>Res.</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>Res.</td>
<td>Reserved</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **NYETMSK**: NYET interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 13 **NAKMSK**: NAK interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 12 **BERRM**: Babble error interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BNAM**: BNA interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 8 **OUTPKTERRM**: Out packet error mask
   0: Masked interrupt
   1: Unmasked interrupt
Bit 7  Reserved, must be kept at reset value.

Bit 6  **B2BSTUPM**: Back-to-back SETUP packets received mask
       Applies to control OUT endpoints only.
       0: Masked interrupt
       1: Unmasked interrupt

Bit 5  **STSPHSRXM**: Status phase received for control write mask
       0: Masked interrupt
       1: Unmasked interrupt

Bit 4  **OTEPM**: OUT token received when endpoint disabled mask. Applies to control OUT
       endpoints only.
       0: Masked interrupt
       1: Unmasked interrupt

Bit 3  **STUPM**: STUPM: SETUP phase done mask. Applies to control endpoints only.
       0: Masked interrupt
       1: Unmasked interrupt

Bit 2  **AHBERRM**: AHB error mask
       0: Masked interrupt
       1: Unmasked interrupt

Bit 1  **EPDM**: Endpoint disabled interrupt mask
       0: Masked interrupt
       1: Unmasked interrupt

Bit 0  **XFRCM**: Transfer completed interrupt mask
       0: Masked interrupt
       1: Unmasked interrupt

**62.14.44  OTG device all endpoints interrupt register (OTG_DAINT)**

Address offset: 0x818
Reset value: 0x0000 0000

When a significant event occurs on an endpoint, a OTG_DAINT register interrupts the
application using the device OUT endpoints interrupt bit or device IN endpoints interrupt bit
of the OTG_GINTSTS register (OEPINT or IEPINT in OTG_GINTSTS, respectively). There
is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits
for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits
are used. Bits in this register are set and cleared when the application sets and clears bits in
the corresponding device endpoint-x interrupt register (OTG_DIEPINTx/OTG_DOEPINTx).

<table>
<thead>
<tr>
<th>OEPINT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IEPINT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>r r r r r r r r r r r r r r</td>
</tr>
</tbody>
</table>
Bits 31:16 **OEPINT[15:0]:** OUT endpoint interrupt bits
   One bit per OUT endpoint:
   Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.

Bits 15:0 **IEPINT[15:0]:** IN endpoint interrupt bits
   One bit per IN endpoint:
   Bit 0 for IN endpoint 0, bit 3 for endpoint 3.

62.14.45 **OTG all endpoints interrupt mask register**
(OTG_DAINTMSK)

Address offset: 0x81C
Reset value: 0x0000 0000

The OTG_DAINTMSK register works with the device endpoint interrupt register to interrupt
the application when an event occurs on a device endpoint. However, the OTG_DAINT
register bit corresponding to that interrupt is still set.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16 **OEPM[15:0]:** OUT EP interrupt mask bits
   One per OUT endpoint:
   Bit 16 for OUT EP 0, bit 19 for OUT EP 3
   0: Masked interrupt
   1: Unmasked interrupt

Bits 15:0 **IEPM[15:0]:** IN EP interrupt mask bits
   One bit per IN endpoint:
   Bit 0 for IN EP 0, bit 3 for IN EP 3
   0: Masked interrupt
   1: Unmasked interrupt
62.14.46 OTG device VBUS discharge time register
(OTG_DVBUSDIS)

Address offset: 0x0828
Reset value: 0x0000 17D7

This register specifies the VBUS discharge time after VBUS pulsing during SRP.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16: Reserved, must be kept at reset value.
Bits 15:0 **VBUSDT[15:0]**: Device VBUS discharge time

Specifies the VBUS discharge time after VBUS pulsing during SRP. This value equals:

\[ \text{VBUS discharge time in PHY clocks} \div 1024 \]

Depending on your VBUS load, this value may need adjusting.

62.14.47 OTG device VBUS pulsing time register
(OTG_DVBUSPULSE)

Address offset: 0x082C
Reset value: 0x0000 05B8

This register specifies the VBUS pulsing time during SRP.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16: Reserved, must be kept at reset value.
Bits 15:0 **DVBUSP[15:0]**: Device VBUS pulsing time. This feature is only relevant to OTG1.3.

Specifies the VBUS pulsing time during SRP. This value equals:

\[ \text{VBUS pulsing time in PHY clocks} \div 1024 \]
### 62.14.48 OTG device threshold control register (OTG_DTHRCTL)

Address offset: 0x0830  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:28</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 27</td>
<td><strong>ARPEN</strong>: Arbiter parking enable</td>
</tr>
<tr>
<td></td>
<td>This bit controls internal DMA arbiter parking for IN endpoints. When thresholding is enabled and this bit is set to one, then the arbiter parks on the IN endpoint for which there is a token received on the USB. This is done to avoid getting into underrun conditions. By default parking is enabled.</td>
</tr>
<tr>
<td>Bit 26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 25:17</td>
<td><strong>RXTHRLEN[8:0]</strong>: Receive threshold length</td>
</tr>
<tr>
<td></td>
<td>This field specifies the receive thresholding size in 32-bit words. This field also specifies the amount of data received on the USB before the core can start transmitting on the AHB. The threshold length has to be at least eight 32-bit words. The recommended value for RXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_GAHBCFG).</td>
</tr>
<tr>
<td>Bit 16</td>
<td><strong>RXTHREN</strong>: Receive threshold enable</td>
</tr>
<tr>
<td></td>
<td>When this bit is set, the core enables thresholding in the receive direction.</td>
</tr>
<tr>
<td>Bits 15:11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 10:2</td>
<td><strong>TXTHRLEN[8:0]</strong>: Transmit threshold length</td>
</tr>
<tr>
<td></td>
<td>This field specifies the transmit thresholding size in 32-bit words. This field specifies the amount of data in bytes to be in the corresponding endpoint transmit FIFO before the core can start transmitting on the USB. The threshold length has to be at least eight 32-bit words. This field controls both isochronous and nonisochronous IN endpoint thresholds. The recommended value for TXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_GAHBCFG).</td>
</tr>
<tr>
<td>Bit 1</td>
<td><strong>ISOTHREN</strong>: ISO IN endpoint threshold enable</td>
</tr>
<tr>
<td></td>
<td>When this bit is set, the core enables thresholding for isochronous IN endpoints.</td>
</tr>
<tr>
<td>Bit 0</td>
<td><strong>NONISOTHREN</strong>: Nonisochronous IN endpoints threshold enable</td>
</tr>
<tr>
<td></td>
<td>When this bit is set, the core enables thresholding for nonisochronous IN endpoints.</td>
</tr>
</tbody>
</table>
62.14.49 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)

Address offset: 0x834
Reset value: 0x0000 0000

This register is used to control the IN endpoint FIFO empty interrupt generation (TXFE_OTG_DIEPINTx).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

INEPTXFEM[15:0]

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 INEPTXFEM[15:0]: IN EP Tx FIFO empty interrupt mask bits
These bits act as mask bits for OTG_DIEPINTx.
TXFE interrupt one bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3
0: Masked interrupt
1: Unmasked interrupt

62.14.50 OTG device each endpoint interrupt register (OTG_DEACHINT)

Address offset: 0x0838
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

OEP1 INT

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 OEP1INT: OUT endpoint 1 interrupt bit
Bits 16:2 Reserved, must be kept at reset value.

Bit 1 IEP1INT: IN endpoint 1 interrupt bit
Bit 0 Reserved, must be kept at reset value.
62.14.51 OTG device each endpoint interrupt mask register (OTG_DEACHINTMSK)

Address offset: 0x083C
Reset value: 0x0000 0000

There is one interrupt bit for endpoint 1 IN and one interrupt bit for endpoint 1 OUT.

| Bit 31:18 | Reserved, must be kept at reset value. |
| Bit 17   | OEP1INTM: OUT endpoint 1 interrupt mask bit |
| Bit 16:2 | Reserved, must be kept at reset value. |
| Bit 1    | IEP1INTM: IN endpoint 1 interrupt mask bit |
| Bit 0    | Reserved, must be kept at reset value. |

62.14.52 OTG device each IN endpoint-1 interrupt mask register (OTG_HS_DIEPEACHMSK1)

Address offset: 0x844
Reset value: 0x0000 0000

This register works with the OTG_DIEPINT1 register to generate a dedicated interrupt OTG_HS_EP1_IN for endpoint #1. The IN endpoint interrupt for a specific status in the OTG_DOEPINT1 register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

| Bit 31:14 | Reserved, must be kept at reset value. |
| Bit 13    | NAKM: NAK interrupt mask |
| 0: Masked interrupt |
| 1: Unmasked interrupt |
| Bit 12:10 | Reserved, must be kept at reset value. |
Bit 9 **BNAM**: BNA interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 8 **TXFURM**: FIFO underrun mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **NEPNE**: IN endpoint NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 Reserved, must be kept at reset value.

Bit 4 **ITTXFEMSK**: IN token received when Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **TOM**: Timeout condition mask (Non-isochronous endpoints)
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERM**: AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM**: Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM**: Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

62.14.53 **OTG device each OUT endpoint-1 interrupt mask register (OTG_HS_DOEPEACHMSK1)**

Address offset: 0x884
Reset value: 0x0000 0000

This register works with the OTG_DOEPI1 register to generate a dedicated interrupt
OTG_HS_EP1_ OUT for endpoint #1. The OUT endpoint interrupt for a specific status in the
OTG_DOEPI1 register can be masked by writing into the corresponding bit in this
register. Status bits are masked by default.
Bits 31:15  Reserved, must be kept at reset value.

Bit 14  **NYETMSK**: NYET interrupt mask
        0: Masked interrupt
        1: Unmasked interrupt

Bit 13  **NAKMSK**: NAK interrupt mask
        0: Masked interrupt
        1: Unmasked interrupt

Bit 12  **BERRM**: Babble error interrupt mask
        0: Masked interrupt
        1: Unmasked interrupt

Bits 11:10  Reserved, must be kept at reset value.

Bit 9  **BNAM**: BNA interrupt mask
        0: Masked interrupt
        1: Unmasked interrupt

Bit 8  **OUTPKTERM**: Out packet error mask
        0: Masked interrupt
        1: Unmasked interrupt

Bit 7  Reserved, must be kept at reset value.

Bit 6  **B2BSTUMP**: Back-to-back SETUP packets received mask
        Applies to control OUT endpoints only.
        0: Masked interrupt
        1: Unmasked interrupt

Bit 5  Reserved, must be kept at reset value.

Bit 4  **OTEPDM**: OUT token received when endpoint disabled mask
        Applies to control OUT endpoints only.
        0: Masked interrupt
        1: Unmasked interrupt

Bit 3  **STUMP**: STUPM: SETUP phase done mask
        Applies to control endpoints only.
        0: Masked interrupt
        1: Unmasked interrupt

Bit 2  **AHBERRM**: AHB error mask
        0: Masked interrupt
        1: Unmasked interrupt

Bit 1  **EPDM**: Endpoint disabled interrupt mask
        0: Masked interrupt
        1: Unmasked interrupt

Bit 0  **XFRCM**: Transfer completed interrupt mask
        0: Masked interrupt
        1: Unmasked interrupt
62.14.54 OTG device IN endpoint x control register (OTG_DIEPCTLx)

Address offset: \(0x900 + 0x20 \times x\), \((x = 0\) to 8\)

Reset value: 0x0000 0000

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

<table>
<thead>
<tr>
<th>EPENA</th>
<th>EPDIS</th>
<th>SODDFRM</th>
<th>SD0PID/SEVNFRM</th>
<th>SNACK</th>
<th>TXFNUM[3:0]</th>
<th>STALL</th>
<th>EFTYP[1:0]</th>
<th>NAKSTS</th>
<th>EO NUM/DPID</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs</td>
<td>rs</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

USBA


<table>
<thead>
<tr>
<th>rw</th>
</tr>
</thead>
</table>

Bit 31 EPENA: Endpoint enable
The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 EPDIS: Endpoint disable
The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 SODDFRM: Set odd frame
Applies to isochronous IN and OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 SD0PID: Set DATA0 PID
Applies to interrupt/bulk IN endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

SEVNFRM: Set even frame
Applies to isochronous IN endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 SNACK: Set NAK
A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 CNAK: Clear NAK
A write to this bit clears the NAK bit for the endpoint.
Bits 25:22 **TXFNUM[3:0]: Tx FIFO number**
These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.

Bit 21 **STALL**: STALL handshake
Applies to non-control, non-isochronous IN endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority.
Only the application can clear this bit, never the core.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]: Endpoint type**
This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS**: NAK status
It indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit:
For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the Tx FIFO.
For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the Tx FIFO.
Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM**: Even/odd frame
Applies to isochronous IN endpoints only.
Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.
0: Even frame
1: Odd frame

**DPID**: Endpoint data PID
Applies to interrupt/bulk IN endpoints only.
Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.
0: DATA0
1: DATA1
Bit 15 **USBAEP**: USB active endpoint  
Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]**: Maximum packet size  
The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

### 62.14.55 OTG device IN endpoint x interrupt register (OTG_DIEPINTx)

Address offset: 0x908 + 0x20 * x, (x = 0 to 8)  
Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 720*. The application must read this register when the IN endpoints interrupt bit of the core interrupt register (IEPINT in OTG_GINTSTS) is set. Before the application can read this register, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td><strong>NAK</strong>: NAK input</td>
</tr>
<tr>
<td>30</td>
<td>The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td><strong>PKTDRPSTS</strong>: Packet dropped status</td>
</tr>
<tr>
<td>27</td>
<td>This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td><strong>BNA</strong>: Buffer not available interrupt</td>
</tr>
<tr>
<td>24</td>
<td>The core generates this interrupt when the descriptor accessed is not ready for the core to process, such as host busy or DMA done. This bit is only valid when Scatter/Gather DMA mode is enabled.</td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td><strong>EPDISD</strong>: EP DISUPPORTED</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td><strong>XFRC</strong>: USB packet transmitted to core</td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td><strong>TOC</strong>: Transaction completed</td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td><strong>AHBERR</strong>: AHB error</td>
</tr>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14</td>
<td><strong>BNA</strong>: Buffer not available interrupt</td>
</tr>
<tr>
<td>13</td>
<td>The core generates this interrupt when the descriptor accessed is not ready for the core to process, such as host busy or DMA done. This bit is only valid when Scatter/Gather DMA mode is enabled.</td>
</tr>
<tr>
<td>12</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>11</td>
<td><strong>PKTDRPSTS</strong>: Packet dropped status</td>
</tr>
<tr>
<td>10</td>
<td>This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.</td>
</tr>
<tr>
<td>9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>8</td>
<td><strong>IN</strong>: IN endpoint</td>
</tr>
<tr>
<td>7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>6</td>
<td><strong>IN</strong>: IN endpoint</td>
</tr>
<tr>
<td>5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>4</td>
<td><strong>ITXFE</strong>: IN Transmit to core</td>
</tr>
<tr>
<td>3</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>2</td>
<td><strong>TXIF</strong>: Transmit FIFO empty</td>
</tr>
<tr>
<td>1</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>0</td>
<td><strong>Res</strong>: Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bit 8  **TXFIFOUDRN**: Transmit Fifo Underrun (TxfifoUndrn)

The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. Dependency: This interrupt is valid only when Thresholding is enabled.

Bit 7  **TXFE**: Transmit FIFO empty

This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the Tx FIFO Empty Level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Bit 6  **INEPNE**: IN endpoint NAK effective

This bit can be cleared when the application clears the IN endpoint NAK by writing to the CNAK bit in OTG_DIEPCTLx. This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.

Bit 5  **INEPNM**: IN token received with EP mismatch

Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 4  **ITTXFE**: IN token received when Tx FIFO is empty

Indicates that an IN token was received when the associated Tx FIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 3  **TOC**: Timeout condition

Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.

Bit 2  **AHBERR**: AHB error

This is generated only in internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.

Bit 1  **EPDISD**: Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application’s request.

Bit 0  **XFRC**: Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.
62.14.56 OTG device IN endpoint 0 transfer size register
(OTG_DIEPTSIZ0)

Address offset: 0x910
Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the device control endpoint 0 control registers (EPENA in OTG_DIEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–3.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 PKTCNT[1:0]: Packet count
Indicates the total number of USB packets that constitute the transfer size amount of data for endpoint 0.
This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 XFRSIZ[6:0]: Transfer size
Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

62.14.57 OTG device IN endpoint x DMA address register
(OTG_DIEPDMAX)

Address offset: 0x914 + 0x20 * x, (x = 0 to 8)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAADDR[31:16]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| DMAADDR[15:0]| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
Bits 31:0 **DMAADDR[31:0]**: DMA Address

This field holds the start address in the external memory from which the data for the endpoint must be fetched. This register is incremented on every AHB transaction.

### 62.14.58 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)

Address offset: 0x918 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0200

This read-only register contains the free space information for the device IN endpoint Tx FIFO.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>DMAADDR[31:0]: DMA Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>INEPTFSAV[15:0]: IN endpoint Tx FIFO space available</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicate the amount of free space available in the endpoint Tx FIFO. Values are in terms of 32-bit words: 0x0: Endpoint Tx FIFO is full 0x1: 1 word available 0x2: 2 words available 0xn: n words available Others: Reserved</td>
</tr>
</tbody>
</table>

### 62.14.59 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)

Address offset: 0x910 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using the endpoint enable bit in the OTG_DIEPCTLx registers (EPENA bit in OTG_DIEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>MCNT[1:0]:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PKTCNT[9:0]:</td>
</tr>
<tr>
<td></td>
<td>XFRSZ[18:16]:</td>
</tr>
<tr>
<td></td>
<td>XFRSZ[15:0]:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **MCNT[1:0]:** Multi count
For periodic IN endpoints, this field indicates the number of packets that must be transmitted per frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.
- 01: 1 packet
- 10: 2 packets
- 11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count
Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.
This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size
This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

### 62.14.60 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)

Address offset: 0xB00
Reset value: 0x0000 8000

This section describes the OTG_DOEPCTL0 register. Nonzero control endpoints use registers for endpoints 1–3.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>r</td>
<td>w</td>
<td>w</td>
<td>rs</td>
<td>rw</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
</tr>
</tbody>
</table>

**Bit 31 EPENA:** Endpoint enable
The application sets this bit to start transmitting data on endpoint 0.
The core clears this bit before setting any of the following interrupts on this endpoint:
- SETUP phase done
- Endpoint disabled
- Transfer completed

**Bit 30 EPDIS:** Endpoint disable
The application cannot disable control OUT endpoint 0.

Bits 29:28 Reserved, must be kept at reset value.
Bit 27 **SNAK**: Set NAK
A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK**: Clear NAK
A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL**: STALL handshake
The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM**: Snoop mode
This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]**: Endpoint type
Hardcoded to 2'b00 for control.

Bit 17 **NAKSTS**: NAK status
Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit, the core stops receiving data, even if there is space in the Rx FIFO to accommodate the incoming packet. Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP**: USB active endpoint
This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ[1:0]**: Maximum packet size
The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN endpoint 0.
00: 64 bytes
01: 32 bytes
10: 16 bytes
11: 8 bytes
62.14.61 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)

Address offset: 0xB08 + 0x20 * x, (x = 0 to 8)
Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in Figure 720. The application must read this register when the OUT endpoints interrupt bit of the OTG_GINTSTS register (OEPINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the OTG_DAINT register to get the exact endpoint number for the OTG_DOEPINTx register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

| Bit 31:16 Reserved, must be kept at reset value. |
| Bit 15 **STPKTRX**: Setup packet received |
| Applicable for control OUT endpoints in only the Buffer DMA Mode. Set by the OTG_HS, this bit indicates that this buffer holds 8 bytes of setup data. There is only one setup packet per buffer. On receiving a setup packet, the OTG_HS closes the buffer and disables the corresponding endpoint after SETUP_COMPLETE status is seen in the Rx FIFO. OTG_HS puts a SETUP_COMPLETE status into the Rx FIFO when it sees the first IN or OUT token after the SETUP packet for that particular endpoint. The application must then re-enable the endpoint to receive any OUT data for the control transfer and reprogram the buffer start address. Because of the above behavior, OTG_HS can receive any number of back to back setup packets and one buffer for every setup packet is used. |

| Bit 14 **NYET**: NYET interrupt |
| This interrupt is generated when a NYET response is transmitted for a non isochronous OUT endpoint. |

| Bit 13 **NAK**: NAK input |
| The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO. |

| Bit 12 **BERR**: Babble error interrupt |
| The core generates this interrupt when babble is received for the endpoint. |

| Bits 11:10 Reserved, must be kept at reset value. |

| Bit 9 **BNA**: Buffer not available interrupt |
| The core generates this interrupt when the descriptor accessed is not ready for the core to process, such as host busy or DMA done. This bit is only valid when Scatter/Gather DMA mode is enabled. |
Bit 8 **OUTPKTERR**: OUT packet error
   This interrupt is asserted when the core detects an overflow or a CRC error for an OUT
   packet. This interrupt is valid only when thresholding is enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUP**: Back-to-back SETUP packets received
   Applies to control OUT endpoint only.
   This bit indicates that the core has received more than three back-to-back SETUP packets
   for this particular endpoint.

Bit 5 **STSPHSRX**: Status phase received for control write
   This interrupt is valid only for control OUT endpoints. This interrupt is generated only after
   OTG_HS has transferred all the data that the host has sent during the data phase of a
   control write transfer, to the system memory buffer. The interrupt indicates to the application
   that the host has switched from data phase to the status phase of a control write transfer.
   The application can use this interrupt to ACK or STALL the status phase, after it has
   decoded the data phase.

Bit 4 **OTEPDIS**: OUT token received when endpoint disabled
   Applies only to control OUT endpoints.
   Indicates that an OUT token was received when the endpoint was not yet enabled. This
   interrupt is asserted on the endpoint for which the OUT token was received.

Bit 3 **STUP**: SETUP phase done
   Applies to control OUT endpoint only. Indicates that the SETUP phase for the control
   endpoint is complete and no more back-to-back SETUP packets were received for the
   current control transfer. On this interrupt, the application can decode the received SETUP
   data packet.

Bit 2 **AHBERR**: AHB error
   This is generated only in internal DMA mode when there is an AHB error during an AHB
   read/write. The application can read the corresponding endpoint DMA address register to
   get the error address.

Bit 1 **EPDISD**: Endpoint disabled interrupt
   This bit indicates that the endpoint is disabled per the application’s request.

Bit 0 **XFRC**: Transfer completed interrupt
   This field indicates that the programmed transfer is complete on the AHB as well as on the
   USB, for this endpoint.
62.14.62 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)

Address offset: 0xB10
Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the OTG_DOEPCTL0 registers (EPENA bit in OTG_DOEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–8.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  Reserved, must be kept at reset value.

Bits 30:29  **STUPCNT[1:0]:** SETUP packet count
This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:20  Reserved, must be kept at reset value.

Bit 19  **PKTCNT:** Packet count
This field is decremented to zero after a packet is written into the Rx FIFO.

Bits 18:7  Reserved, must be kept at reset value.

Bits 6:0  **XFRSIZ[6:0]:** Transfer size
Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.
### 62.14.63 OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx)

Address offset: 0xB14 + 0x20 * x, (x = 0 to 8)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>bits 31:16</th>
<th>DMAADDR[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:0 DMAADDR[31:0]: DMA Address
This field holds the start address in the external memory from which the data for the endpoint must be fetched. This register is incremented on every AHB transaction.

### 62.14.64 OTG device OUT endpoint x control register (OTG_DOEPCTLx)

Address offset: 0xB00 + 0x20 * x, (x = 1 to 8)
Reset value: 0x0000 0000

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>rs rs w w w w w w w rw/rs rw rw rw r r</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

rw rw rw rw rw rw rw rw rw rw rw rw
Bit 31 **EPENA**: Endpoint enable
Applies to IN and OUT endpoints.
The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
- SETUP phase done
- Endpoint disabled
- Transfer completed

Bit 30 **EPDIS**: Endpoint disable
The application sets this bit to stop transmitting/receiving data on an endpoint, even before
the transfer for that endpoint is complete. The application must wait for the endpoint
disabled interrupt before treating the endpoint as disabled. The core clears this bit before
setting the endpoint disabled interrupt. The application must set this bit only if endpoint
enable is already set for this endpoint.

Bit 29 **SD1PID**: Set DATA1 PID
Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the endpoint
data PID (DPID) field in this register to DATA1.

**SODDFRM**: Set odd frame
Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd
frame (EONUM) field to odd frame.

Bit 28 **SD0PID**: Set DATA0 PID
Applies to interrupt/bulk OUT endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

**SEVNFRM**: Set even frame
Applies to isochronous OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK**: Set NAK
A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an
endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt,
or after a SETUP is received on the endpoint.

Bit 26 **CNAK**: Clear NAK
A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL**: STALL handshake
Applies to non-control, non-isochronous OUT endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK
bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes
priority. Only the application can clear this bit, never the core.
Applies to control endpoints only (access type is rs).
The application can only set this bit, and the core clears it, when a SETUP token is received
for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit,
the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to
SETUP data packets with an ACK handshake.

Bit 20 **SNPM**: Snoop mode
This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check
the correctness of OUT packets before transferring them to application memory.
Bits 19:18 **EPTYP[1:0]**: Endpoint type
This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS**: NAK status
Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit:
The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.
Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM**: Even/odd frame
Applies to isochronous IN and OUT endpoints only.
Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.
0: Even frame
1: Odd frame

**DPID**: Endpoint data PID
Applies to interrupt/bulk OUT endpoints only.
Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.
0: DATA0
1: DATA1

Bit 15 **USBAEP**: USB active endpoint
Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]**: Maximum packet size
The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.
62.14.65  OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)

Address offset: 0xB10 + 0x20 * x, (x = 1 to 8)
Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using endpoint enable bit of the OTG_DOEPCTLx registers (EPENA bit in OTG_DOEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r/rw</td>
<td>r/rw</td>
<td>rw</td>
</tr>
<tr>
<td>31</td>
<td>15</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>30-29</td>
<td>RXDPID[1:0]: Received data PID</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Applies to isochronous OUT endpoints only.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This is the data PID received in the last packet for this endpoint.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00: DATA0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01: DATA2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10: DATA1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11: MDATA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28-19</td>
<td>STUPCNT[1:0]: SETUP packet count</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Applies to control OUT endpoints only.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This field specifies the number of back-to-back SETUP data packets the endpoint can receive.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01: 1 packet</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10: 2 packets</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11: 3 packets</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18-0</td>
<td>PKTCNT[9:0]: Packet count</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This field is decremented every time a packet (maximum size or short packet) is written to the Rx FIFO.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>XFRSIZ[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>rw</td>
</tr>
<tr>
<td>14-13</td>
<td>rw</td>
</tr>
<tr>
<td>12-9</td>
<td>rw</td>
</tr>
<tr>
<td>8-7</td>
<td>rw</td>
</tr>
<tr>
<td>6-5</td>
<td>rw</td>
</tr>
<tr>
<td>4-3</td>
<td>rw</td>
</tr>
<tr>
<td>2-1</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31  Reserved, must be kept at reset value.

Bits 30:29  RXDPID[1:0]: Received data PID

This is the data PID received in the last packet for this endpoint.

00: DATA0
01: DATA2
10: DATA1
11: MDATA

STUPCNT[1:0]: SETUP packet count

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.

01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:19  PKTCNT[9:0]: Packet count

This field indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.

This field is decremented every time a packet (maximum size or short packet) is written to the Rx FIFO.

Bits 18:0  XFRSIZ[18:0]: Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.
62.14.66  OTG power and clock gating control register (OTG_PCGCCTL)

Address offset: 0xE00
Reset value: 0x200B 8000

This register is available in host and device modes.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>rw</td>
<td>f</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:8**  Reserved, must be kept at reset value.

**Bit 7**  **SUSP**: Deep Sleep
This bit indicates that the PHY is in Deep Sleep when in L1 state.

**Bit 6**  **PHYSLEEP**: PHY in Sleep
This bit indicates that the PHY is in the Sleep state.

**Bit 5**  **ENL1GTG**: Enable sleep clock gating
When this bit is set, core internal clock gating is enabled in Sleep state if the core cannot assert utmi_l1_suspend_n. When this bit is not set, the PHY clock is not gated in Sleep state.

**Bit 4**  **PHYSUSP**: PHY suspended
Indicates that the PHY has been suspended. This bit is updated once the PHY is suspended after the application has set the STPPCLK bit.

**Bits 3:2**  Reserved, must be kept at reset value.

**Bit 1**  **GATEHCLK**: Gate HCLK
The application sets this bit to gate HCLK to modules other than the AHB Slave and Master and wakeup logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.

**Bit 0**  **STPPCLK**: Stop PHY clock
The application sets this bit to stop the PHY clock when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.
2640/2974
0

OTG_
GINTSTS
LPMINT
PTXFE
HCINT
HPRTINT
RSTDET
DATAFSUSP
IPXFR/INCOMPISOOUT
IISOIXFR
OEPINT
IEPINT

Reset value
0
0
0
0
0
1
0
0
0
0
0
0
0
0

OTG_
GINTMSK

LPMINTM

PTXFEM

HCIM

PRTIM

RSTDETM

FSUSPM

IPXFRM/IISOOXFRM

IISOIXFRM

OEPINT

IEPINT

Reset value

0

0

0

0

0

0

0

0

0

0

0

0

0

0

RM0455 Rev 3
Res.

0

0

0

0

0

0

0
OTGINT
MMIS
CMOD

0
0
1
0
0
0
0
0

OTGINT

MMISM

Res.

AVALOEN
VBVALOVA
VBVALOEN
SRQ
SRQSCS

0
0

Res.

0
0
0

Res.
Res.
Res.

DMAEN

Res.

Res.
Res.
Res.
Res.
Res.
SEDET
Res.
Res.

0

1
CSRST

BVALOEN
AVALOVAL

0

0

PSRST

BVALOVAL

0

TXFELVL

0

0

FCRST

Res.

HNPRQ
HNGSCS

0

SRSSCHG

0

HNSSCHG

0

Res.

DHNPEN
HSHNPEN

0

PTXFELVL

EHEN

HBSTLEN

0
0

GINTMSK

Res.

Res.

0

Res.

Res.

Res.

Res.

0

0

PHYSEL

DBCT
CIDSTS

Res.

0

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

ASVLD
HNGDET

Res.

BSVLD
ADTOCHG

Res.

OTGVER

Res.
DBCDNE

Res.

Res.

0

-

Res.

0

SOF

0

RXFLVL

0

SOFM

0

RXFLVLM

0

NPTXFE

0

NPTXFEM

0

GINAKEFF

0

GONAKEFF

SRPCAP

0

GINAKEFFM

RXFFLSH

0

HNPCAP

1

GONAKEFFM

Res.

PHYLPC

Reset value

Res.

Res.

TXFNUM
TXFFLSH

ESUSP

0

ESUSPM

0

Res.

Res.

Res.

1

Res.

USBSUSP

0

USBSUSPM

Res.

0

Res.

USBRST

0

USBRST

Res.

0
TRDT

Res.

ENUMDNE

0

ENUMDNEM

ULPIFSL

0

EOPF

ULPIAR

0

Res.

Res.
CURMOD

Res.

Res.

1

ISOODRP

ULPICSM

0

Res.

Res.

Res.

Res.

0

EOPFM

ULPIEVBUSD

0

Res.

Res.

Res.

Res.

0

ISOODRPM

ULPIEVBUSI

0
Res.

TSDPS

0

Res.

Res.

Res.

Res.

Res.

Res.

Res.

0

0

Res.

PCCI

0

Res.

Res.

Res.

Res.

Res.

31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Res.
Res.

OTG_
GOTGCTL

0

0

Res.

PTCI

0

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Register
name

0

0

Res.

ULPIIPD

Res.

Res.
0

Res.

Res.

Res.

Res.

Res.

Reset value

Res.

1
Res.

Res.

Reset value

Res.

FHMOD

Res.

Reset value

DISCINT

OTG_
GUSBCFG

Res.

OTG_
GRSTCTL

CIDSCHG

OTG_
GAHBCFG
Res.

OTG_
GOTGINT

Res.

0

DISCINT

0x018
0

CIDSCHGM

0x014
FDMOD

0x010
Reset value

AHBIDL

0x00C

DMAREQ

0x008

SRQINT

0x004

WKUPINT

0x000

WUIM

Offset

SRQIM

USB on-the-go high-speed (OTG_HS)
RM0455

62.14.67 OTG_HS register map
The table below gives the USB OTG register map and reset values.
Table 502. OTG_HS register map and reset values

0

0

TOCAL

0
0
0

0
0
0

0

0

0

0

0

0

0


Table 502. OTG_HS register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01C</td>
<td>OTG_GRXSTSR (Device mode)</td>
<td>0x020</td>
<td>OTG_GRXSTSP (Device mode)</td>
<td>0x024</td>
<td>OTG_GRXFSIZ</td>
<td>0x028</td>
<td>OTG_HNPTXFSIZ/OTG_DIEPTXF0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x02C</td>
<td>OTG_HNPTXSTS</td>
<td>0x038</td>
<td>OTG_GCCFG</td>
<td>0x03C</td>
<td>OTG_CID</td>
<td>0x054</td>
<td>OTG_GLPMCFG</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x100</td>
<td>OTG_HPTXFSIZ</td>
<td>0x104</td>
<td>OTG_DIEPTXF1</td>
<td>0x108</td>
<td>OTG_DIEPTXF2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01C</td>
<td>OTG_GRXSTSR (Device mode)</td>
<td>0x020</td>
<td>OTG_GRXSTSP (Device mode)</td>
<td>0x024</td>
<td>OTG_GRXFSIZ</td>
<td>0x028</td>
<td>OTG_HNPTXFSIZ/OTG_DIEPTXF0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x02C</td>
<td>OTG_HNPTXSTS</td>
<td>0x038</td>
<td>OTG_GCCFG</td>
<td>0x03C</td>
<td>OTG_CID</td>
<td>0x054</td>
<td>OTG_GLPMCFG</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x100</td>
<td>OTG_HPTXFSIZ</td>
<td>0x104</td>
<td>OTG_DIEPTXF1</td>
<td>0x108</td>
<td>OTG_DIEPTXF2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reset value in each register varies depending on the mode (Device or Host). The table includes registers such as `FRMNUM`, `PKTSTS`, `D PID`, `BCNT`, and `EPNUM` which are used for managing frames, packets, device ID, block count, and endpoint number respectively. The reset values are indicated for both Device and Host modes.
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x120</td>
<td>OTG_DIEPTXF7</td>
<td>INEPTXFD</td>
<td>INEPTXSA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x400</td>
<td>OTG_HCFG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x404</td>
<td>OTG_HFIR</td>
<td>RLDCFR</td>
<td>FRIVL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 1 1 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x408</td>
<td>OTG_HFNUM</td>
<td>FTREM</td>
<td>FRNUN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x410</td>
<td>OTG_HPTXSTS</td>
<td>PTXQTOP</td>
<td>PTXQSAV</td>
<td>PTXFSAVL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x414</td>
<td>OTG_HAINT</td>
<td>HAINM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x418</td>
<td>OTG_HAINTMSK</td>
<td>HAINTM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x41C</td>
<td>OTG_HFLBADDR</td>
<td>HFLBADDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x440</td>
<td>OTG_HPRT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x500</td>
<td>OTG_HCCCHAR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x504</td>
<td>OTG_HCSPLT0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 502. OTG_HS register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x508  | OTG_HCINT0    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x508  | OTG_HCINT0    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x50C  | OTG_HCINTMSK0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x510  | OTG_HCTSIZ0   | DOPNG | DPID | PKTCNT | XFRSIZ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x510  | OTG_HCTSIZSZ0 | DOPNG | PID | NTD | SCHED_INFO |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x514  | OTG_HCDMA0    | DMAADDR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x514  | OTG_HCDMA5G0  | DMASG[31:3] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x51C  | OTG_HCDMAB0   | HCDMAB |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x520  | OTG_HCHAR1    | CHENA | CHDIS | ODDFRM | DAD | MCNT | EPTYP | LSDEV | EPDIR | EPNUM | MPSIZ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x524  | OTG_HCSPLT1   | SPLDUIEN | COMPLSP | XAC | TPS | HUBADDR | PRTADDR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x528  | OTG_HCINT1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x528  | OTG_HCINT1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
## OTG_HS Register Map and Reset Values (continued)

<p>| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x52C  | OTG_HCINTMSK1 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x530  | OTG_HCTSIZ1   | DODP | PID | PKTCNT | XFRSIZ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x530  | OTG_HCTSIZS1G1 | DODP | PID | NTD | SCHED_INFO |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x534  | OTG_HCDMA1    | DMAADDR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x534  | OTG_HCDMASG1  | DOPNG | DAD | PKTCNT | XFRSIZ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x53C  | OTG_HCDMAB    | HCDMAB |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x660  | OTG_HCCHAR1   | CHENA | CHDIS | ODDFRM | DAD | MCNT | EPTYP | LSDEV | EPDIR | EPNUM | MPSIZ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x664  | OTG_HCSPLT1   | SPLITEN | COMPSPLIT | XAC | TPO | HUBADDR | PRTADDR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x668  | OTG_HCINT11   | DTERM | FNMORM | BBERRM | TXERR | ACKM | NAKM | STALLM | CHIC | XFRM |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x66C  | OTG_HCINTMSK11 | DTERM | FNMORM | BBERRM | TXERR | ACKM | NAKM | STALLM | CHIC | XFRM |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x670  | OTG_HCTSIZ1   | DODP | PID | PKTCNT | XFRSIZ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |</p>
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>O15</th>
<th>O14</th>
<th>O13</th>
<th>O12</th>
<th>O11</th>
<th>O10</th>
<th>O9</th>
<th>O8</th>
<th>O7</th>
<th>O6</th>
<th>O5</th>
<th>O4</th>
<th>O3</th>
<th>O2</th>
<th>O1</th>
<th>O0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x670</td>
<td>OTG_HCTSIZSG11</td>
<td>PID</td>
<td>NTD</td>
<td>SCHED_INFO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x674</td>
<td>OTG_HCDMA11</td>
<td>DMAADDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x674</td>
<td>OTG_HCDMASG11</td>
<td>DMSG[31:3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x67C</td>
<td>OTG_HCDMAB11</td>
<td>HCDMAB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x6E0</td>
<td>OTG_HCCHAR15</td>
<td>CHENA</td>
<td>CHDIS</td>
<td>ODDFRM</td>
<td>DAD</td>
<td>MCNT</td>
<td>EPTYP</td>
<td>LSDEV</td>
<td>EPOK</td>
<td>EPNUM</td>
<td>MPSIZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x6E4</td>
<td>OTG_HCSPLT15</td>
<td>SPURERN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x6E8</td>
<td>OTG_HCINT15</td>
<td>DTERRE</td>
<td>FRMORM</td>
<td>BBERRM</td>
<td>TXERR</td>
<td>ACK</td>
<td>NAK</td>
<td>STALL</td>
<td>CHH</td>
<td>XFR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x6EC</td>
<td>OTG_HCINTMSK15</td>
<td>DTERRE</td>
<td>FRMORM</td>
<td>BBERRM</td>
<td>TXERR</td>
<td>NYET</td>
<td>ACK</td>
<td>NAK</td>
<td>NAKM</td>
<td>STALL</td>
<td>CHH</td>
<td>XFR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x6F0</td>
<td>OTG_HCTSIZ15</td>
<td>DOPNG</td>
<td>PTKCNT</td>
<td>XFRSIZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x6F0</td>
<td>OTG_HCTSIZSG15</td>
<td>DOPNG</td>
<td>NTD</td>
<td>SCHED_INFO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x6F4</td>
<td>OTG_HCDMA15</td>
<td>DMAADDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>0xF4</td>
<td>OTG_HCDMASG15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFC</td>
<td>OTG_HCDMAB15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x000</td>
<td>OTG_DCFG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>OTG_DCTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>OTG_DSTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>OTG_DIEPMSK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>OTG_DOEPMASK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x018</td>
<td>OTG_DAIMT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01C</td>
<td>OTG_DAIMTMSK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x28</td>
<td>OTG_DVBUSDIS</td>
<td>VBUSDT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2C</td>
<td>OTG_DVBUSPulse</td>
<td>DVBUSP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 502. OTG_HS register map and reset values (continued)
### Table 502. OTG_HS register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x830  | OTG_DTHRCTL  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x834  | OTG_DIEPEMPMSK |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x838  | OTG_DEACHINT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x83C  | OTG_DEACHINTMSK |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x844  | OTG_HS_DOEPEACHMSK1 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x848  | OTG_HS_DOEPEACHMSK1 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x900  | OTG_DIEPCTL0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x904  | OTG_DIEPINT0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x908  | OTG_DIEPTSIZ0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x910  | OTG_DIEPDMA |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x914  | OTG_DIEPDMA |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DMAADDR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Table 502. OTG_HS register map and reset values (continued)

| Offset | Register name   | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x918  | DTXFSTS0       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x920  | DIEPCTL1       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | MPSIZ |     |     |     |     |     |     |     |     |     |
|        |                | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x928  | DIEPINT1       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x930  | DIEPSIZ1       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x938  | DIEFSTS1       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x940  | DIEPCTL2       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | MPSIZ |     |     |     |     |     |     |     |     |     |     |
|        |                | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
Table 502. OTG_HS register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x9E0  | OTG_DIEPCTL7  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x9E8  | OTG_DIEPINT7  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x9F0  | OTG_DIEPTSIZ7 | MCNT | PKTCNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x9F8  | OTG_DTXFSTS7  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xB00  | OTG_DOEPCCTL0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xB08  | OTG_DOEPINT0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xB10  | OTG_DOEPTSIZ0 |     |     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xB14  | OTG_DOEPDMA0  |     |     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
### Table 502. OTG_HS register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xB20  | OTG_DOEPCtl1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xB28  | OTG_DOEPINT1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xB30  | OTG_DOEPTSIZ1|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xB34  | OTG_DOEPDMA1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xC00  | OTG_DOEPCtl8  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xC08  | OTG_DOEPINT8  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xC10  | OTG_DOEPTSIZ8|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xC14  | OTG_DOEPDMA8  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xE00  | OTG_PGCCTL    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Refer to *Section 2.3 on page 128* for the register boundary addresses.
62.15 OTG_HS programming model

62.15.1 Core initialization

The application must perform the core initialization sequence. If the cable is connected during power-up, the current mode of operation bit in the OTG_GINTSTS (CMOD bit in OTG_GINTSTS) reflects the mode. The OTG_HS controller enters host mode when an “A” plug is connected or device mode when a “B” plug is connected.

This section explains the initialization of the OTG_HS controller after power-on. The application must follow the initialization sequence irrespective of host or device mode operation. All core global registers are initialized according to the core’s configuration:

1. Program the following fields in the OTG_GAHBCFG register:
   - Global interrupt mask bit GINTMSK = 1
   - Rx FIFO non-empty (RXFLVL bit in OTG_GINTSTS)
   - Periodic Tx FIFO empty level

2. Program the following fields in the OTG_GUSBCFG register:
   - HNP capable bit
   - SRP capable bit
   - OTG_HS timeout calibration field
   - USB turnaround time field

3. The software must unmask the following bits in the OTG_GINTMSK register:
   - OTG interrupt mask
   - Mode mismatch interrupt mask

4. The software can read the CMOD bit in OTG_GINTSTS to determine whether the OTG_HS controller is operating in host or device mode.

62.15.2 Host initialization

To initialize the core as host, the application must perform the following steps:
1. Program the HPRTINT in the OTG_GINTMSK register to unmask
2. Program the OTG_HCFG register to select full-speed host
3. Program the PPWR bit in OTG_HPRT to 1. This drives $V_{BUS}$ on the USB.
4. Wait for the PCDET interrupt in OTG_HPRT0. This indicates that a device is connecting to the port.
5. Program the PRST bit in OTG_HPRT to 1. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program the PRST bit in OTG_HPRT to 0.
8. Wait for the PENCHNG interrupt in OTG_HPRT.
9. Read the PSPD bit in OTG_HPRT to get the enumerated speed.
10. Program the HFIR register with a value corresponding to the selected PHY clock 1
11. Program the FSLSPCS field in the OTG_HCFG register following the speed of the device detected in step 9. If FSLSPCS has been changed a port reset must be performed.
12. Program the OTG_GRXFSIZ register to select the size of the receive FIFO.
13. Program the OTG_HNPTXFSIZ register to select the size and the start address of the Non-periodic transmit FIFO for non-periodic transactions.
14. Program the OTG_HPTXFSIZ register to select the size and start address of the periodic transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel.

62.15.3 Device initialization

The application must perform the following steps to initialize the core as a device on power-up or after a mode change from host to device.

1. Program the following fields in the OTG_DCFG register:
   - DESCDMA
   - Device speed
   - Non-zero-length status OUT handshake
   - Periodic Frame Interval
2. Program the Device threshold control register. This is required only if you are using DMA mode and you are planning to enable thresholding.
3. Clear the DCTL.SDIS bit. The core issues a connect after this bit is cleared.
4. Program the OTG_GINTMSK register to unmask the following interrupts:
   - USB reset
   - Enumeration done
   - Early suspend
   - USB suspend
   - SOF
5. Wait for the USBRST interrupt in OTG_GINTSTS. It indicates that a reset has been detected on the USB that lasts for about 10 ms on receiving this interrupt.
6. Wait for the ENUMDNDE interrupt in OTG_GINTSTS. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the OTG_DSTS
register to determine the enumeration speed and perform the steps listed in "Endpoint initialization on enumeration completion on page 2686." At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

### 62.15.4 DMA mode

The OTG host uses the AHB master interface to fetch the transmit packet data (AHB to USB) and receive the data update (USB to AHB). The AHB master uses the programmed DMA address (OTG_HCDMAx register in host mode and OTG_DIEPDMAx/OTG_DOEPDMAx register in peripheral mode) to access the data buffers.

**Scatter/Gather DMA mode**

In Scatter/Gather DMA mode, the core implements a true scatter/gather memory distribution in which data buffers are scattered over the system memory. However, the descriptors themselves are continuous. Each channel memory structure is implemented as a contiguous list of descriptors; each descriptor points to a data buffer of predefined size. In addition to the buffer pointer (a 32-bit word), the descriptor also has a status quadlet (32-bit word). When the list is implemented as a ring buffer, the list processor switches to the first element of the list when it encounters last bit. All channels (control, bulk, interrupt, and isochronous) implement these structures in memory. When Scatter/Gather DMA is enabled in device mode, OTG_DIEPDMAx and OTG_DOEPDMAx registers are used to access the base descriptor.

### 62.15.5 Host programming model

**Channel initialization**

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps:

1. Program the OTG_GINTMSK register to unmask the following:
   - Channel interrupt
   - Non-periodic transmit FIFO empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
   - Non-periodic transmit FIFO half-empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).

2. Program the OTG_HAINTMSK register to unmask the selected channels' interrupts.

3. Program the OTG_HCINTMSK register to unmask the transaction-related interrupts of interest given in the host channel interrupt register.

4. Program the selected channel’s OTG_HCTSIZx register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).

5. Program the OTG_HCCHARx register of the selected channel with the device’s endpoint characteristics, such as type, speed, direction, and so forth. (The channel can...
be enabled by setting the channel enable bit to 1 only when the application is ready to transmit or receive any packet).

7. Program the selected channels in the OTG_HCSPLTx register(s) with the hub and port addresses (split transactions only).

8. Program the selected channels in the OTG_HCDMAx register(s) with the buffer start address (DMA transactions only).

Halting a channel

The application can disable any channel by programming the OTG_HCCHARx register with the CHDIS and CHENA bits set to 1. This enables the OTG_HS host to flush the posted requests (if any) and generates a channel halted interrupt. The application must wait for the CHH interrupt in OTG_HCINTx before reallocating the channel for other transactions. The OTG_HS host does not interrupt the transaction that has already been started on the USB.

To disable a channel in DMA mode operation, the application does not need to check for space in the request queue. The OTG_HS host checks for space to write the disable request on the disabled channel’s turn during arbitration. Meanwhile, all posted requests are dropped from the request queue when the CHDIS bit in OTG_HCCHARx is set to 1.

Before disabling a channel, the application must ensure that there is at least one free space available in the non-periodic request queue (when disabling a non-periodic channel) or the periodic request queue (when disabling a periodic channel). The application can simply flush the posted requests when the request queue is full (before disabling the channel), by programming the OTG_HCCHARx register with the CHDIS bit set to 1, and the CHENA bit cleared to 0.

The application is expected to disable a channel on any of the following conditions:

1. When an STALL, TXERR, BBERR or DTERR interrupt in OTG_HCINTx is received for an IN or OUT channel. The application must be able to receive other interrupts (DTERR, Nak, data, TXERR) for the same channel before receiving the halt.

2. When an XFRC interrupt in OTG_HCINTx is received during a non periodic IN transfer or high-bandwidth interrupt IN transfer.

3. When a DISCINT (disconnect device) interrupt in OTG_GINTSTS is received. (The application is expected to disable all enabled channels).

4. When the application aborts a transfer before normal completion.

Ping protocol

When the OTG_HS host operates in high speed, the application must initiate the ping protocol when communicating with high-speed bulk or control (data and status stage) OUT endpoints. The application must initiate the ping protocol when it receives a NAK/NYET/TXERR interrupt. When the OTG_HS host receives one of the above responses, it does not continue any transaction for a specific endpoint, drops all posted or fetched OUT requests (from the request queue), and flushes the corresponding data (from the transmit FIFO). This is valid in slave mode only. In Slave mode, the application can send a ping token either by setting the DOPING bit in OTG_HCTSIZx before enabling the channel or by just writing the OTG_HCTSIZx register with the DOPING bit set when the channel is already enabled. This enables the OTG_HS host to write a ping request entry to the request queue. The application must wait for the response to the ping token (a NAK, ACK, or TXERR interrupt) before continuing the transaction or sending another ping token. The application can continue the data transaction only after receiving an ACK from the OUT endpoint for the requested ping. In DMA mode operation, the application does not need to
set the DOPING bit in OTG_HCTSIZx for a NAK/NYET response in case of bulk/control OUT. The OTG_HS host automatically sets the DOPING bit in OTG_HCTSIZx, and issues the ping tokens for bulk/control OUT. The OTG_HS host continues sending ping tokens until it receives an ACK, and then switches automatically to the data transaction.

**Operational model**

The application must initialize a channel before communicating to the connected device. This section explains the sequence of operation to be performed for different types of USB transactions.

- **Writing the transmit FIFO**

  The OTG_HS host automatically writes an entry (OUT request) to the periodic/non-periodic request queue, along with the last 32-bit word write of a packet. The application must ensure that at least one free space is available in the periodic/non-periodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in 32-bit words. If the packet size is non-32-bit word aligned, the application must use padding. The OTG_HS host determines the actual packet size based on the programmed maximum packet size and transfer size.

![Figure 721. Transmit FIFO write task](image)

- **Reading the receive FIFO**

  The application must ignore all packet statuses other than IN data packet (bx0010).
• **Bulk and control OUT/SETUP transactions**
  A typical bulk or control OUT/SETUP pipelined transaction-level operation is shown in Figure 723. See channel 1 (ch_1). Two bulk OUT packets are transmitted. A control SETUP transaction operates in the same way but has only one packet. The assumptions are:
  – The application is attempting to send two maximum-packet-size packets (transfer size = 1,024 bytes).
  – The non-periodic transmit FIFO can hold two packets (1 KB for HS).
  – The non-periodic request queue depth = 4.

• **Normal bulk and control OUT/SETUP operations**
  The sequence of operations in (channel 1) is as follows:
  1. Initialize channel 1
  2. Write the first packet for channel 1
  3. Along with the last word write, the core writes an entry to the non-periodic request queue
  4. As soon as the non-periodic queue becomes non-empty, the core attempts to send an OUT token in the current frame
  5. Write the second (last) packet for channel 1
  6. The core generates the XFRC interrupt as soon as the last transaction is completed successfully
  7. In response to the XFRC interrupt, de-allocate the channel for other transfers
  8. Handling non-ACK responses
Figure 723. Normal bulk/control OUT/SETUP

1. The grayed elements are not relevant in the context of this figure.
The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions is shown in the following code samples.

- **Interrupt service routine for bulk/control OUT/SETUP and bulk/control IN transactions**
  a) Bulk/control OUT/SETUP

        Unmask (NAK/ TXERR/ STALL/ XFRC)
        if (XFRC)
            {
                Reset Error Count
                Mask ACK
                De-allocate Channel
            }
        else if (STALL)
            {
                Transfer Done = 1
                Unmask CHH
                Disable Channel
            }
        else if (NAK or TXERR )
            {
                Rewind Buffer Pointers
                Unmask CHH
                Disable Channel
                if (TXERR)
                    {
                        Increment Error Count
                        Unmask ACK
                    }
                else
                    {
                        Reset Error Count
                    }
            }
        else if (CHH)
            {
                Mask CHH
                if (Transfer Done or (Error_count == 3))
                    {
                        De-allocate Channel
                    }
                else
                    {
                        Re-initialize Channel
                    }
            }
else if (ACK)
{
    Reset Error Count
    Mask ACK
}

The application is expected to write the data packets into the transmit FIFO when the space is available in the transmit FIFO and the request queue. The application can make use of the NPTXFE interrupt in OTG_GINTSTS to find the transmit FIFO space.

b) Bulk/control IN

Unmask (TXERR/XFRC/BBERR/STALL/DTErr)
if (XFRC)
{
    Reset Error Count
    Unmask CHH
    Disable Channel
    Reset Error Count
    Mask ACK
}
else if (TXERR or BBERR or STALL)
{
    Unmask CHH
    Disable Channel
    if (TXERR)
    {
        Increment Error Count
        Unmask ACK
    }
}
else if (CHH)
{
    Mask CHH
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel
    }
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}
else if (DTERR)
{
    Reset Error Count
}

The application is expected to write the requests as and when the request queue space is available and until the XFRC interrupt is received.

- **Bulk and control IN transactions**
  A typical bulk or control IN pipelined transaction-level operation is shown in Figure 724. See channel 2 (ch_2). The assumptions are:
  - The application is attempting to receive two maximum-packet-size packets (transfer size = 1 024 bytes).
  - The receive FIFO can contain at least one maximum-packet-size packet and two status words per packet (520 bytes for HS).
  - The non-periodic request queue depth = 4.
Figure 724. Bulk/control IN transactions

1. The grayed elements are not relevant in the context of this figure.

Non-Periodic Request Queue
Assume that this queue can hold 4 entries.
The sequence of operations is as follows:

1. Initialize channel 2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the non-periodic request queue.
3. The core attempts to send an IN token after completing the current OUT transaction.
4. The core generates an RXFLVL interrupt as soon as the received packet is written to the receive FIFO.
5. In response to the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. Following this, unmask the RXFLVL interrupt.
6. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO.
7. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in OTG_GRXSTSR ≠ 0b0010).
8. The core generates the XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, disable the channel and stop writing the OTG_HCCHAR2 register for further requests. The core writes a channel disable request to the non-periodic request queue as soon as the OTG_HCCHAR2 register is written.
10. The core generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.
11. Read and ignore the receive packet status.
12. The core generates a CHH interrupt as soon as the halt status is popped from the receive FIFO.
13. In response to the CHH interrupt, de-allocate the channel for other transfers.
14. Handling non-ACK responses

- **Control transactions**
  Setup, data, and status stages of a control transfer must be performed as three separate transfers. setup-, data- or status-stage OUT transactions are performed similarly to the bulk OUT transactions explained previously. Data- or status-stage IN transactions are performed similarly to the bulk IN transactions explained previously. For all three stages, the application is expected to set the EPTYP field in OTG_HCCHAR1 to control. During the setup stage, the application is expected to set the PID field in OTG_HCTSIZ1 to SETUP.

- **Interrupt OUT transactions**
  A typical interrupt OUT operation is shown in Figure 725. The assumptions are:
  - The application is attempting to send one packet in every frame (up to 1 maximum packet size), starting with the odd frame (transfer size = 1 024 bytes)
  - The periodic transmit FIFO can hold one packet (1 KB)
  - Periodic request queue depth = 4
  The sequence of operations is as follows:
1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG_HS host writes an entry to the periodic request queue.
4. The OTG_HS host attempts to send an OUT token in the next (odd) frame.
5. The OTG_HS host generates an XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.
Figure 725. Normal interrupt OUT

1. The grayed elements are not relevant in the context of this figure.

- Interrupt service routine for interrupt OUT/IN transactions
  a) Interrupt OUT
      Unmask (NAK/TXERR/STALL/XFRC/FRMOR)
if (XFRC)
{
    Reset Error Count
    Mask ACK
    De-allocate Channel
}
else
    if (STALL or FRMOR)
    {
        Mask ACK
        Unmask CHH
        Disable Channel
        if (STALL)
        {
            Transfer Done = 1
        }
    }
else
    if (NAK or TXERR)
    {
        Rewind Buffer Pointers
        Reset Error Count
        Mask ACK
        Unmask CHH
        Disable Channel
    }
else
    if (CHH)
    {
        Mask CHH
        if (Transfer Done or (Error_count == 3))
        {
            De-allocate Channel
        }
    }
else
    if (ACK)
    {
        Reset Error Count
        Mask ACK
    }
The application uses the NPTXFE interrupt in OTG_GINTSTS to find the transmit FIFO space.

Interrupt IN
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTER)
if (XFRC)
{
    Reset Error Count
    Mask ACK
    if (OTG_HCTSIZx.PKTCNT == 0)
    {
        De-allocate Channel
    }
else
    {
        Transfer Done = 1
        Unmask CHH
        Disable Channel
    }
else
    if (STALL or FRMOR or NAK or DTERR or BBERR)
    {
        Mask ACK
        Unmask CHH
        Disable Channel
        if (STALL or BBERR)
        {
            Reset Error Count
            Transfer Done = 1
        }
else
        if (!FRMOR)
        {
            Reset Error Count
        }
    }
else
    if (TXERR)
    {
        Increment Error Count
        Unmask ACK
        Unmask CHH
        Disable Channel
    }
else
if (CHH)
{
    Mask CHH
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    Re-initialize Channel (in next b_interval - 1 /Frame)
}
else
if (ACK)
{
    Reset Error Count
    Mask ACK
}

• Interrupt IN transactions
The assumptions are:
– The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd (transfer size = 1 024 bytes).
– The receive FIFO can hold at least one maximum-packet-size packet and two status words per packet (1 031 bytes).
– Periodic request queue depth = 4.

• Normal interrupt IN operation
The sequence of operations is as follows:
1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG_HS host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG_HS host attempts to send an IN token in the next (odd) frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG_HS host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask after reading the entire packet.
7. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If the PKTCNT bit in OTG_HCTSIZ2 is not equal to 0, disable the channel before re-
initializing the channel for the next transfer, if any). If PKTCNT bit in OTG_HCTSIZ2 = 0, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.
1. The grayed elements are not relevant in the context of this figure.

- **Isochronous OUT transactions**

A typical isochronous OUT operation is shown in Figure 726. The assumptions are:

- The application is attempting to send one packet every frame (up to 1 maximum
packet size), starting with an odd frame. (transfer size = 1 024 bytes).

– The periodic transmit FIFO can hold one packet (1 KB).
– Periodic request queue depth = 4.

The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG_HS host writes an entry to the periodic request queue.
4. The OTG_HS host attempts to send the OUT token in the next frame (odd).
5. The OTG_HS host generates the XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.
7. Handling non-ACK responses
Figure 727. Isochronous OUT transactions

1. The grayed elements are not relevant in the context of this figure.

   • Interrupt service routine for isochronous OUT/IN transactions

   Code sample: isochronous OUT

   Unmask (FRMOR/XFRC)

   if (XFRC)

2672/2974  RM0455 Rev 3
{ De-allocate Channel } 
else 
  if (FRMOR) 
    { 
      Unmask CHH 
      Disable Channel 
    } 
else 
  if (CHH) 
    { 
      Mask CHH 
      De-allocate Channel 
    } 

Code sample: Isochronous IN
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR)
  { 
    if (XFRC and (OTG_HCTSIZx.PKTCNT == 0)) 
      { 
        Reset Error Count 
        De-allocate Channel 
      } 
    else 
      { 
        Unmask CHH 
        Disable Channel 
      } 
  } 
else 
  if (TXERR or BBERR)
    { 
      Increment Error Count 
      Unmask CHH 
      Disable Channel 
    } 
else 
  if (CHH)
    { 
      Mask CHH 
      if (Transfer Done or (Error_count == 3)) 
        { 
          De-allocate Channel 
        }
    }
else
{
  Re-initialize Channel
}

- Isochronous IN transactions

  The assumptions are:
  - The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame (transfer size = 1 024 bytes).
  - The receive FIFO can hold at least one maximum-packet-size packet and two status word per packet (1 031 bytes).
  - Periodic request queue depth = 4.

  The sequence of operations is as follows:

  1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
  2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
  3. The OTG_HS host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
  4. The OTG_HS host attempts to send an IN token in the next odd frame.
  5. As soon as the IN packet is received and written to the receive FIFO, the OTG_HS host generates an RXFLVL interrupt.
  6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask it after reading the entire packet.
  7. The core generates an RXFLVL interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS bit in OTG_GRXSTSR ≠ 0b0010).
  8. The core generates an XFRC interrupt as soon as the receive packet status is read.
  9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If PKTCNT ≠ 0 in OTG_HCTSIZ2, disable the channel before re-initializing the channel for the next transfer, if any. If PKTCNT = 0 in OTG_HCTSIZ2, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.
Figure 728. Isochronous IN transactions

1. The grayed elements are not relevant in the context of this figure.

- **Selecting the queue depth**

  Choose the periodic and non-periodic request queue depths carefully to match the number of periodic/non-periodic endpoints accessed.

  The non-periodic request queue depth affects the performance of non-periodic
transfers. The deeper the queue (along with sufficient FIFO size), the more often the core is able to pipeline non-periodic transfers. If the queue size is small, the core is able to put in new requests only when the queue space is freed up.

The core’s periodic request queue depth is critical to perform periodic transfers as scheduled. Select the periodic queue depth, based on the number of periodic transfers scheduled in a microframe. If the periodic request queue depth is smaller than the periodic transfers scheduled in a microframe, a frame overrun condition occurs.

- **Handling babble conditions**
  OTG_HS controller handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more data than the maximum packet size for the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

  When OTG_HS controller detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already written data in the Rx buffer and generates a Babble interrupt to the application.

  When OTG_HS controller detects a port babble, it flushes the Rx FIFO and disables the port. The core then generates a port disabled interrupt (HPRTINT in OTG_GINTSTS, PENCHNG in OTG_HPRT). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the port disabled interrupt) by checking POCA in OTG_HPRT, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

- **Bulk and control OUT/SETUP transactions in DMA mode**
  The sequence of operations is as follows:
  1. Initialize and enable channel 1 as explained in *Section: Channel initialization*.
  2. The OTG_HS host starts fetching the first packet as soon as the channel is enabled. For internal DMA mode, the OTG_HS host uses the programmed DMA address to fetch the packet.
  3. After fetching the last 32-bit word of the second (last) packet, the OTG_HS host masks channel 1 internally for further arbitration.
  4. The OTG_HS host generates a CHH interrupt as soon as the last packet is sent.
  5. In response to the CHH interrupt, de-allocate the channel for other transfers.
Figure 729. Normal bulk/control OUT/SETUP transactions - DMA

- **NAK and NYET handling with internal DMA:**
  1. The OTG_HS host sends a bulk OUT transaction.
  2. The device responds with NAK or NYET.
  3. If the application has unmasked NAK or NYET, the core generates the corresponding interrupt(s) to the application. The application is not required to service these interrupts, since the core takes care of rewinding the buffer pointers and re-initializing the Channel without application intervention.
  4. The core automatically issues a ping token.
  5. When the device returns an ACK, the core continues with the transfer. Optionally, the application can utilize these interrupts, in which case the NAK or NYET interrupt is masked by the application.
The core does not generate a separate interrupt when NAK or NYET is received by the host functionality.

- **Bulk and control IN transactions in DMA mode**
  The sequence of operations is as follows:
  1. Initialize and enable the used channel (channel x) as explained in *Section : Channel initialization*.
  2. The OTG_HS host writes an IN request to the request queue as soon as the channel receives the grant from the arbiter (arbitration is performed in a round-robin fashion).
  3. The OTG_HS host starts writing the received data to the system memory as soon as the last byte is received with no errors.
  4. When the last packet is received, the OTG_HS host sets an internal flag to remove any extra IN requests from the request queue.
  5. The OTG_HS host flushes the extra requests.
  6. The final request to disable channel x is written to the request queue. At this point, channel 2 is internally masked for further arbitration.
  7. The OTG_HS host generates the CHH interrupt as soon as the disable request comes to the top of the queue.
  8. In response to the CHH interrupt, de-allocate the channel for other transfers.
• **Interrupt OUT transactions in DMA mode**

1. Initialize and enable channel x as explained in Section: Channel initialization.
2. The OTG_HS host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last 32-bit word fetch. In high-bandwidth transfers, the OTG_HS host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.
3. The OTG_HS host attempts to send the OUT token at the beginning of the next odd frame/micro-frame.
4. After successfully transmitting the packet, the OTG_HS host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

**Figure 7.31. Normal interrupt OUT transactions - DMA mode**

- **Interrupt IN transactions in DMA mode**
  The sequence of operations (channelx) is as follows:
  1. Initialize and enable channel x as explained in Section: Channel initialization.
  2. The OTG_HS host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the OTG_HS host writes consecutive writes up to MC times.
3. The OTG_HS host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.

4. As soon the packet is received and written to the receive FIFO, the OTG_HS host generates a CHH interrupt.

5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

**Figure 732. Normal interrupt IN transactions - DMA mode**

- Isochronous OUT transactions in DMA mode
  1. Initialize and enable channel x as explained in Section : Channel initialization.
  2. The OTG_HS host starts fetching the first packet as soon as the channel is enabled, and writes the OUT request along with the last 32-bit word fetch. In high-bandwidth
transfers, the OTG_HS host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.

3. The OTG_HS host attempts to send an OUT token at the beginning of the next (odd) frame/micro-frame.

4. After successfully transmitting the packet, the OTG_HS host generates a CHH interrupt.

5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 733. Normal isochronous OUT transaction - DMA mode

- Isochronous IN transactions in DMA mode
  The sequence of operations ((channel x) is as follows:
  1. Initialize and enable channel x as explained in Section: Channel initialization.
  2. The OTG_HS host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the OTG_HS host performs consecutive write operations up to MC times.
3. The OTG_HS host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.
4. As soon the packet is received and written to the receive FIFO, the OTG_HS host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 734. Normal isochronous IN transactions - DMA mode

- **Bulk and control OUT/SETUP split transactions in DMA mode**

  The sequence of operations in (channel x) is as follows:

  1. Initialize and enable channel x for start split as explained in *Section: Channel initialization*.

  2. The OTG_HS host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last 32-bit word fetch.

  3. After successfully transmitting start split, the OTG_HS host generates the CHH interrupt.

  4. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT1 to send the complete split.
5. After successfully transmitting complete split, the OTG_HS host generates the CHH interrupt.
6. In response to the CHH interrupt, de-allocate the channel.

- **Bulk/control IN split transactions in DMA mode**
  The sequence of operations (channel x) is as follows:
  1. Initialize and enable channel x as explained in *Section: Channel initialization.*
  2. The OTG_HS host writes the start split request to the nonperiodic request after getting the grant from the arbiter. The OTG_HS host masks the channel x internally for the arbitration after writing the request.
  3. As soon as the IN token is transmitted, the OTG_HS host generates the CHH interrupt.
  4. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 and re-enable the channel to send the complete split token. This unmasks channel x for arbitration.
  5. The OTG_HS host writes the complete split request to the nonperiodic request after receiving the grant from the arbiter.
  6. The OTG_HS host starts writing the packet to the system memory after receiving the packet successfully.
  7. As soon as the received packet is written to the system memory, the OTG_HS host generates a CHH interrupt.
  8. In response to the CHH interrupt, de-allocate the channel.

- **Interrupt OUT split transactions in DMA mode**
  The sequence of operations in (channel x) is as follows:
  1. Initialize and enable channel 1 for start split as explained in *Section: Channel initialization.* The application must set the ODDFRM bit in OTG_HCCHAR1.
  2. The OTG_HS host starts reading the packet.
  3. The OTG_HS host attempts to send the start split transaction.
  4. After successfully transmitting the start split, the OTG_HS host generates the CHH interrupt.
  5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSSPLT1 to send the complete split.
  6. After successfully completing the complete split transaction, the OTG_HS host generates the CHH interrupt.
  7. In response to CHH interrupt, de-allocate the channel.

- **Interrupt IN split transactions in DMA mode**
  The sequence of operations in (channel x) is as follows:
  1. Initialize and enable channel x for start split as explained in *Section: Channel initialization.*
  2. The OTG_HS host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.
  3. The OTG_HS host attempts to send the start split IN token at the beginning of the next odd micro-frame.
  4. The OTG_HS host generates the CHH interrupt after successfully transmitting the start split IN token.
  5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSSPLT2 to send the complete split.
6. As soon as the packet is received successfully, the OTG_HS host starts writing the data to the system memory.

7. The OTG_HS host generates the CHH interrupt after transferring the received data to the system memory.

8. In response to the CHH interrupt, de-allocate or reinitialize the channel for the next start split.

- Isochronous OUT split transactions in DMA mode

  The sequence of operations (channel x) is as follows:

  1. Initialize and enable channel x for start split (begin) as explained in Section : Channel initialization. The application must set the ODDFRM bit in OTG_HCCHAR1. Program the MPS field.

  2. The OTG_HS host starts reading the packet.

  3. After successfully transmitting the start split (begin), the OTG_HS host generates the CHH interrupt.

  4. In response to the CHH interrupt, reinitialize the registers to send the start split (end).

  5. After successfully transmitting the start split (end), the OTG_HS host generates a CHH interrupt.

  6. In response to the CHH interrupt, de-allocate the channel.

- Isochronous IN split transactions in DMA mode

  The sequence of operations (channel x) is as follows:

  1. Initialize and enable channel x for start split as explained in Section : Channel initialization.

  2. The OTG_HS host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.

  3. The OTG_HS host attempts to send the start split IN token at the beginning of the next odd micro-frame.

  4. The OTG_HS host generates the CHH interrupt after successfully transmitting the start split IN token.

  5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 to send the complete split.

  6. As soon as the packet is received successfully, the OTG_HS host starts writing the data to the system memory.

    The OTG_HS host generates the CHH interrupt after transferring the received data to the system memory. In response to the CHH interrupt, de-allocate the channel or reinitialize the channel for the next start split.
62.15.6 Device programming model

Endpoint initialization on USB reset

1. Set the NAK bit for all OUT endpoints
   - SNAK = 1 in OTG_DOEPCTLx (for all OUT endpoints)
2. Unmask the following interrupt bits
   - INEP0 = 1 in OTG_DAINTMSK (control 0 IN endpoint)
   - OUTEP0 = 1 in OTG_DAINTMSK (control 0 OUT endpoint)
   - STUPM = 1 in OTG_DOEPMSK
   - XFRCM = 1 in OTG_DOEPMSK
   - XFRCM = 1 in OTG_DIEPMSK
   - TOM = 1 in OTG_DIEPMSK
3. Set up the data FIFO RAM for each of the FIFOs
   - Program the OTG_GRXFSIZ register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 words (for the status of the control OUT data packet) + 10 words (for setup packets).
   - Program the OTG_DIEPTXF0 register (depending on the FIFO number chosen) to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.
4. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
   - STUPCNT = 3 in OTG_DOEPTSIZ0 (to receive up to 3 back-to-back SETUP packets)
5. For USB OTG_HS in DMA mode, the OTG_DOEPDMA0 register should have a valid memory address to store any SETUP packets received.

At this point, all initialization required to receive SETUP packets is done.

Endpoint initialization on enumeration completion

1. On the Enumeration Done interrupt (ENUMDNE in OTG_GINTSTS), read the OTG_DSTS register to determine the enumeration speed.
2. Program the MPSIZ field in OTG_DIEPCTL0 to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.
3. For USB OTG_HS in DMA mode, program the OTG_DOEPCTL0 register to enable control OUT endpoint 0, to receive a SETUP packet.

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

Endpoint initialization on SetAddress command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the OTG_DCFG register with the device address received in the SetAddress command
2. Program the core to send out a status IN packet
Endpoint initialization on SetConfiguration/SetInterface command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.

2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.

3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.

4. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_DAINTMSK register.

5. Set up the data FIFO RAM for each FIFO.

6. After all required endpoints are configured; the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

Endpoint activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
   - Maximum packet size
   - USB active endpoint = 1
   - Endpoint start data toggle (for interrupt and bulk endpoints)
   - Endpoint type
   - Tx FIFO number

2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

Endpoint deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB active endpoint bit in the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).

2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, which results in a timeout on the USB.

Note: The application must meet the following conditions to set up the device core to handle traffic:
NPTXFEM and RXFLVLM in the OTG_GINTMSK register must be cleared.
Operational model

SETUP and OUT data transfers:

This section describes the internal data flow and application-level operations during data OUT transfers and SETUP transactions.

- Packet read

This section describes how to read packets (OUT data and SETUP packets) from the receive FIFO.

1. On catching an RXFLVL interrupt (OTG_GINTSTS register), the application must read the receive status pop register (OTG_GRXSTSP).

2. The application can mask the RXFLVL interrupt (in OTG_GINTSTS) by writing to RXFLVLM = 0 (in OTG_GINTMSK), until it has read the packet from the receive FIFO.

3. If the received packet's byte count is not 0, the byte count amount of data is popped from the receive data FIFO and stored in memory. If the received packet byte count is 0, no data is popped from the receive data FIFO.

4. The receive status readout of the packet of FIFO indicates one of the following:
   a) Global OUT NAK pattern:
      PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = (0x0), DPID = (0b00).
      These data indicate that the global OUT NAK bit has taken effect.
   b) SETUP packet pattern:
      PKTSTS = SETUP, BCNT = 0x008, EPNUM = Control EP Num, DPID = DATA0. These data indicate that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.
   c) Setup stage done pattern:
      PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num, DPID = (0b00).
      These data indicate that the setup stage for the specified endpoint has completed and the data stage has started. After this entry is popped from the receive FIFO, the core asserts a setup interrupt on the specified control OUT endpoint.
   d) Data OUT packet pattern:
      PKTSTS = DataOUT, BCNT = size of the received data OUT packet (0 ≤ BCNT ≤ 1 024), EPNUM = EPNUM on which the packet was received, DPID = Actual Data PID.
   e) Data transfer completed pattern:
      PKTSTS = Data OUT transfer done, BCNT = 0x0, EPNUM = OUT EP Num on which the data transfer is complete, DPID = (0b00).
      These data indicate that an OUT data transfer for the specified OUT endpoint has completed. After this entry is popped from the receive FIFO, the core asserts a transfer completed interrupt on the specified OUT endpoint.

5. After the data payload is popped from the receive FIFO, the RXFLVL interrupt (OTG_GINTSTS) must be unmasked.

6. Steps 1–5 are repeated every time the application detects assertion of the interrupt line due to RXFLVL in OTG_GINTSTS. Reading an empty receive FIFO can result in undefined core behavior.

Figure 735 provides a flowchart of the above procedure.
SETUP transactions

This section describes how the core handles SETUP packets and the application’s sequence for handling SETUP transactions.

- **Application requirements**

1. To receive a SETUP packet, the STUPCNT field (OTG_DOEPTSIZx) in a control OUT endpoint must be programmed to a non-zero value. When the application programs the STUPCNT field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status and EPENA bit setting in OTG_DOEPCTLx. The STUPCNT field is decremented every time the control endpoint receives a SETUP packet. If the STUPCNT field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the STUPCNT field, but the application may not be able to determine the correct number of SETUP packets received in the setup stage of a control transfer.
   - STUPCNT = 3 in OTG_DOEPTSIZx

2. The application must always allocate some extra space in the receive data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
   - The space to be reserved is 10 words. Three words are required for the first SETUP packet, 1 word is required for the setup stage done word and 6 words are required to store two extra SETUP packets among all control endpoints.
   - 3 words per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (setup packet pattern). The core reserves this space in the
receive data FIFO to write SETUP data only, and never uses this space for data packets.

3. The application must read the 2 words of the SETUP packet from the receive FIFO.
4. The application must read and discard the setup stage done word from the receive FIFO.

**Internal data flow**

1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint’s NAK and STALL bit settings.
   - The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
2. For every SETUP packet received on the USB, 3 words of data are written to the receive FIFO, and the STUPCNT field is decremented by 1.
   - The first word contains control information used internally by the core
   - The second word contains the first 4 bytes of the SETUP command
   - The third word contains the last 4 bytes of the SETUP command
3. When the setup stage changes to a data IN/OUT stage, the core writes an entry (setup stage done word) to the receive FIFO, indicating the completion of the setup stage.
4. On the AHB side, SETUP packets are emptied by the application.
5. When the application pops the setup stage done word from the receive FIFO, the core interrupts the application with an STUP interrupt (OTG_DOEPINTx), indicating it can process the received SETUP packet.
6. The core clears the endpoint enable bit for control OUT endpoints.

**Application programming sequence**

1. Program the OTG_DOEPTSIZx register.
   - STUPCNT = 3
2. Wait for the RXFLVL interrupt (OTG_GINTSTS) and empty the data packets from the receive FIFO.
3. Assertion of the STUP interrupt (OTG_DOEPINTx) marks a successful completion of the SETUP data transfer.
   - On this interrupt, the application must read the OTG_DOEPTSIZx register to determine the number of SETUP packets received and process the last received SETUP packet.
Handling more than three back-to-back SETUP packets

Per the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP packets a host can send to the same endpoint. When this condition occurs, the OTG_HS controller generates an interrupt (B2BSTUP in OTG_DOEPINTx).

Setting the global OUT NAK

Internal data flow:

1. When the application sets the Global OUT NAK (SGONAK bit in OTG_DCTL), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets.
2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern.
3. When the application pops the Global OUT NAK pattern word from the receive FIFO, the core sets the GONAKEFF interrupt (OTG_GINTSTS).
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the SGONAK bit in OTG_DCTL.

Application programming sequence:
1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field:
   – SGONAK = 1 in OTG_DCTL
2. Wait for the assertion of the GONAKEFF interrupt in OTG_GINTSTS. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set SGONAK in OTG_DCTL and before the core asserts the GONAKEFF interrupt (OTG_GINTSTS).
4. The application can temporarily mask this interrupt by writing to the GONAKEFFM bit in the OTG_GINTMSK register.
   – GONAKEFFM = 0 in the OTG_GINTMSK register
5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the SGONAK bit in OTG_DCTL. This also clears the GONAKEFF interrupt (OTG_GINTSTS).
   – CGONAK = 1 in OTG_DCTL
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
   – GONAKEFFM = 1 in OTG_GINTMSK

- Disabling an OUT endpoint
  The application must use this sequence to disable an OUT endpoint that it has enabled.
  Application programming sequence:
  1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core.
     – SGONAK = 1 in OTG_DCTL
  2. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
  3. Disable the required OUT endpoint by programming the following fields:
     – EPDIS = 1 in OTG_DOEPCTLx
     – SNAK = 1 in OTG_DOEPCTLx
  4. Wait for the EPDISD interrupt (OTG_DOEPINTx), which indicates that the OUT endpoint is completely disabled. When the EPDISD interrupt is asserted, the core also clears the following bits:
     – EPDIS = 0 in OTG_DOEPCTLx
     – EPENA = 0 in OTG_DOEPCTLx
  5. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled OUT endpoints.
     – SGONAK = 0 in OTG_DCTL

- Transfer Stop Programming for OUT endpoints
  The application must use the following programing sequence to stop any transfers (because of an interrupt from the host, typically a reset).
  Sequence of operations:
1. Enable all OUT endpoints by setting
   – EPENA = 1 in all OTG_HS_DOEPCTLx registers.
2. Flush the RxFIFO as follows
   – Poll OTG_HS_GRSTCTL_AHBIDL until it is 1. This indicates that AHB master is idle.
   – Perform read modify write operation on OTG_HS_GRSTCTL.RXFFLSH =1
   – Poll OTG_HS_GRSTCTL_RXFFLSH until it is 0, but also using a timeout of less than 10 milli-seconds (corresponds to minimum reset signaling duration). If 0 is seen before the timeout, then the RxFIFO flush is successful. If at the moment the timeout occurs, there is still a 1, (this may be due to a packet on EP0 coming from the host) then go back (once only) to the previous step (“Perform read modify write operation”).
3. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core, according to the instructions in “Setting the global OUT NAK on page 2691”. This ensures that data in the RxFIFO is sent to the application successfully. Set SGONAK = 1 in OTG_HS_DCTL
4. Wait for the GONAKEFF interrupt (OTG_HS_GINTSTS)
5. Disable all active OUT endpoints by programming the following register bits:
   – EPDIS = 1 in registers OTG_HS_DOEPCTLx
   – SNAK = 1 in registers OTG_HS_DOEPCTLx
6. Wait for the EPDIS interrupt in OTG_HS_DOEPINTx for each OUT endpoint programmed in the previous step. The EPDIS interrupt in OTG_HS_DOEPINTx indicates that the corresponding OUT endpoint is completely disabled. When the EPDIS interrupt is asserted, the following bits are cleared:
   – EPENA = 0 in registers OTG_HS_DOEPCTLx
   – EPDIS = 0 in registers OTG_HS_DOEPCTLx
   – SNAK = 0 in registers OTG_HS_DOEPCTLx

- **Generic non-isochronous OUT data transfers**

  This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

  Application requirements:
1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer.

2. For OUT transfers, the transfer size field in the endpoint’s transfer size register must be a multiple of the maximum packet size of the endpoint, adjusted to the word boundary.
   - \( \text{transfer size[EPNUM]} = n \times (\text{MPSIZ[EPNUM]} + 4 - (\text{MPSIZ[EPNUM]} \mod 4)) \)
   - \( \text{packet count[EPNUM]} = n \)
   - \( n > 0 \)

3. On any OUT endpoint interrupt, the application must read the endpoint’s transfer size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
   - 
     \[
     \text{Payload size in memory} = \text{application programmed initial transfer size} - \text{core updated final transfer size}
     \]
   - 
     \[
     \text{Number of USB packets in which this payload was received} = \text{application programmed initial packet count} - \text{core updated final packet count}
     \]

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.

2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the packet count field for that endpoint by 1.
   - OUT data packets received with bad data CRC are flushed from the receive FIFO automatically.
   - After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, resends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
   - If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
   - In all the above three cases, the packet count is not decremented because no data are written to the receive FIFO.

3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.

4. After the data are written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.

5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.

6. The OUT data transfer completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions:
   - The transfer size is 0 and the packet count is 0
   - The last OUT data packet written to the receive FIFO is a short packet (\(0 \leq \text{packet size} < \text{maximum packet size}\))
7. When either the application pops this entry (OUT data transfer completed), a transfer completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count.
2. Program the OTG_DOEPCTLx register with the endpoint characteristics, and set the EPENA and CNAK bits.
   - EPENA = 1 in OTG_DOEPCTLx
   - CNAK = 1 in OTG_DOEPCTLx
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO.
   - This step can be repeated many times, depending on the transfer size.
4. Asserting the XFRC interrupt (OTG_DOEPINTx) marks a successful completion of the non-isochronous OUT data transfer.
5. Read the OTG_DOEPTSIZx register to determine the size of the received data payload.

- **Generic isochronous OUT data transfer**

This section describes a regular isochronous OUT data transfer.

Application requirements:

1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers.
2. For isochronous OUT data transfers, the transfer size and packet count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
3. The application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (EOPF interrupt in OTG_GINTSTS).
4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the EOPF (OTG_GINTSTS) and before the SOF (OTG_GINTSTS).

Internal data flow:

1. The internal data flow for isochronous OUT endpoints is the same as that for non-isochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the endpoint enable and clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core receives data on an isochronous OUT endpoint in a particular frame only if the following condition is met:
   - EONUM (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
3. When the application completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the RXDPID field in OTG_DOEPTSIZx with the data PID of the last isochronous OUT data packet read from the receive FIFO.

Application programming sequence:
1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count
2. Program the OTG_DOEPCTLx register with the endpoint characteristics and set the endpoint enable, ClearNAK, and Even/Odd frame bits.
   - EPENA = 1
   - CNAK = 1
   - EONUM = (0: Even/1: Odd)
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO
   - This step can be repeated many times, depending on the transfer size.
4. The assertion of the XFRC interrupt (in OTG_DOEPINTx) marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory are good.
5. This interrupt cannot always be detected for isochronous OUT transfers. Instead, the application can detect the INCOMPISOOUT interrupt in OTG_GINTSTS.
6. Read the OTG_DOEPTSIZx register to determine the size of the received transfer and to determine the validity of the data received in the frame. The application must treat the data received in memory as valid only if one of the following conditions is met:
   - RXDPID = DATA0 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 1
   - RXDPID = DATA1 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 2
   - RXDPID = D2 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 3
   - The number of USB packets in which this payload was received = Application programmed initial packet count – core updated final packet count
   The application can discard invalid data packets.

- **Incomplete isochronous OUT data transfers**

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal data flow:
1. For isochronous OUT endpoints, the XFRC interrupt (in OTG_DOEPINTx) may not always be asserted. If the core drops isochronous OUT data packets, the application could fail to detect the XFRC interrupt (OTG_DOEPINTx) under the following circumstances:
   - When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data
   - When the isochronous OUT data packet is received with CRC errors
   - When the isochronous OUT token received by the core is corrupted
   - When the application is very slow in reading the data from the receive FIFO
2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the incomplete isochronous OUT data interrupt (INCOMPISOOUT in OTG_GINTSTS), indicating that an XFRC interrupt (in OTG_DOEPINTx) is not asserted on at least one of the isochronous OUT endpoints. At
this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remain in progress on this endpoint on the USB.

Application programming sequence:

1. Asserting the INCOMPISOOUT interrupt (OTG_GINTSTS) indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.

2. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
   - When all data are emptied from the receive FIFO, the application can detect the XFRC interrupt (OTG_DOEPINTx). In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame.

3. When it receives an INCOMPISOOUT interrupt (in OTG_GINTSTS), the application must read the control registers of all isochronous OUT endpoints (OTG_DOEPCTLx) to determine which endpoints had an incomplete transfer in the current microframe. An endpoint transfer is incomplete if both the following conditions are met:
   - EONUM bit (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
   - EPENA = 1 (in OTG_DOEPCTLx)

4. The previous step must be performed before the SOF interrupt (in OTG_GINTSTS) is detected, to ensure that the current frame number is not changed.

5. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the EPDIS bit in OTG_DOEPCTLx.

6. Wait for the EPDISD interrupt (in OTG_DOEPINTx) and enable the endpoint to receive new data in the next frame.
   - Because the core can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving bad isochronous data.

- **Stalling a non-isochronous OUT endpoint**

This section describes how the application can stall a non-isochronous endpoint.

1. Put the core in the Global OUT NAK mode.

2. Disable the required endpoint
   - When disabling the endpoint, instead of setting the SNAK bit in OTG_DOEPCTL, set STALL = 1 (in OTG_DOEPCTL).
     The STALL bit always takes precedence over the NAK bit.

3. When the application is ready to end the STALL handshake for the endpoint, the STALL bit (in OTG_DOEPCTLx) must be cleared.

4. If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

**Examples**

This section describes and depicts some fundamental transfer types and scenarios.

- Bulk OUT transaction
Figure 737 depicts the reception of a single Bulk OUT data packet from the USB to the AHB and describes the events involved in the process.

**Figure 737. Bulk OUT transaction**

After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting CNAK = 1 and EPENA = 1 (in OTG_DOEPCTLx), and setting a suitable XFRSIZ and PKTCNT in the OTG_DOEPTSIZx register.

1. host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the Rx FIFO because space is available there.
3. After writing the complete packet in the Rx FIFO, the core then asserts the RXFLVL interrupt (in OTG_GINTSTS).
4. On receiving the PKTCNT number of USB packets, the core internally sets the NAK bit for this endpoint to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the Rx FIFO.
6. When the application has read all the data (equivalent to XFRSIZ), the core generates an XFRC interrupt (in OTG_DOEPINTx).
7. The application processes the interrupt and uses the setting of the XFRC interrupt bit (in OTG_DOEPINTx) to determine that the intended transfer is complete.

**IN data transfers**

- **Packet write**
  
  This section describes how the application writes data packets to the endpoint FIFO when dedicated transmit FIFOs are enabled.
1. The application can either choose the polling or the interrupt mode.
   - In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   - In interrupt mode, the application waits for the TXFE interrupt (in OTG_DIEPINTx) and then reads the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   - To write a single non-zero length data packet, there must be space to write the entire packet in the data FIFO.
   - To write zero length packet, the application must not look at the FIFO space.

2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. Typically, the application must do a read modify write on the OTG_DIEPCTLx register to avoid modifying the contents of the register, except for setting the endpoint enable bit.

The application can write multiple packets for the same endpoint into the transmit FIFO, if space is available. For periodic IN endpoints, the application must write packets only for one microframe. It can write packets for the next periodic transaction only after getting transfer complete for the previous transaction.

- Setting IN endpoint NAK

Internal data flow:
1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint’s transmit FIFO.

2. Non-isochronous IN tokens receive a NAK handshake reply
   - Isochronous IN tokens receive a zero-data-length packet reply

3. The core asserts the INEPNE (IN endpoint NAK effective) interrupt in OTG_DIEPINTx in response to the SNAK bit in OTG_DIEPCTLx.

4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the CNAK bit in OTG_DIEPCTLx.

Application programming sequence:
1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
   - SNAK = 1 in OTG_DIEPCTLx

2. Wait for assertion of the INEPNE interrupt in OTG_DIEPINTx. This interrupt indicates that the core has stopped transmitting data on the endpoint.

3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.

4. The application can mask this interrupt temporarily by writing to the INEPNEM bit in OTG_DIEPMSK.
   - INEPNEM = 0 in OTG_DIEPMSK

5. To exit endpoint NAK mode, the application must clear the NAK status bit (NAKSTS) in OTG_DIEPCTLx. This also clears the INEPNE interrupt (in OTG_DIEPINTx).
6. If the application masked this interrupt earlier, it must be unmasked as follows:
   – INEPNEM = 1 in OTG_DIEPMSK

• IN endpoint disable

Use the following sequence to disable a specific IN endpoint that has been previously enabled.

Application programming sequence:
1. The application must stop writing data on the AHB for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode.
   – SNAK = 1 in OTG_DIEPCTLx
3. Wait for the INEPNE interrupt in OTG_DIEPINTx.
4. Set the following bits in the OTG_DIEPCTLx register for the endpoint that must be disabled.
   – EPDIS = 1 in OTG_DIEPCTLx
   – SNAK = 1 in OTG_DIEPCTLx
5. Assertion of the EPDISD interrupt in OTG_DIEPINTx indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits:
   – EPENA = 0 in OTG_DIEPCTLx
   – EPDIS = 0 in OTG_DIEPCTLx
6. The application must read the OTG_DIEPTSIZx register for the periodic IN EP, to calculate how much data on the endpoint were transmitted on the USB.
7. The application must flush the data in the endpoint transmit FIFO, by setting the following fields in the OTG_GRSTCTL register:
   – TXFNUM (in OTG_GRSTCTL) = Endpoint transmit FIFO number
   – TXFFLSH in (OTG_GRSTCTL) = 1

The application must poll the OTG_GRSTCTL register, until the TXFFLSH bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

• Transfer Stop Programming for IN endpoints

The application must use the following programing sequence to stop any transfers (because of an interrupt from the host, typically a reset).
**Sequence of operations:**

1. Disable the IN endpoint by setting:
   - EPDIS = 1 in all OTG_HS_DIEPCTLx registers

2. Wait for the EPDIS interrupt in OTG_HS_DIEPINTx, which indicates that the IN endpoint is completely disabled. When the EPDIS interrupt is asserted the following bits are cleared:
   - EPDIS = 0 in OTG_HS_DIEPCTLx
   - EPENA = 0 in OTG_HS_DIEPCTLx

3. Flush the TxFIFO by programming the following bits:
   - TXFFLSH = 1 in OTG_HS_GRSTCTL
   - TXFNUM = “FIFO number specific to endpoint” in OTG_HS_GRSTCTL

4. The application can start polling till TXFFLSH in OTG_HS_GRSTCTL is cleared. When this bit is cleared, it ensures that there is no data left in the Tx FIFO.

- **Generic non-periodic IN data transfers**

**Application requirements:**

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer are part of a single buffer.

2. For IN transfers, the transfer size field in the endpoint transfer size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
   - To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
     Transfer size[EPNUM] = \(x \times MPSIZ[EPNUM] + sp\)
     If \((sp > 0)\), then packet count[EPNUM] = \(x + 1\).
     Otherwise, packet count[EPNUM] = \(x\)
   - To transmit a single zero-length data packet:
     Transfer size[EPNUM] = 0
     Packet count[EPNUM] = 1
   - To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer into two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.
     - First transfer: transfer size[EPNUM] = \(x \times MPSIZ[epnum]\); packet count = \(n\);
     - Second transfer: transfer size[EPNUM] = 0; packet count = 1;

3. Once an endpoint is enabled for data transfers, the core updates the transfer size register. At the end of the IN transfer, the application must read the transfer size register to determine how much data posted in the transmit FIFO have already been sent on the USB.

4. Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
   - Data transmitted on USB = (application-programmed initial packet count – core updated final packet count) \(\times MPSIZ[EPNUM]\)
   - Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)
Internal data flow:
1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the transmit FIFO for the endpoint.
3. Every time a packet is written into the transmit FIFO by the application, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory by the application, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the “number of packets in FIFO” count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data are written to the transmit FIFO, the core reads them out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a timeout.
5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the packet count field.
6. If there are no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates an “IN token received when Tx FIFO is empty” (ITTXFE) interrupt for the endpoint, provided that the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. The core internally rewinds the FIFO pointers and no timeout interrupt is generated.
8. When the transfer size is 0 and the packet count is 0, the transfer complete (XFRC) interrupt for the endpoint is generated and the endpoint enable is cleared.

Application programming sequence:
1. Program the OTG_DIEPTSIZx register with the transfer size and corresponding packet count.
2. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA (endpoint enable) bits.
3. When transmitting non-zero length data packet, the application must poll the OTG_DTXFSTSx register (where x is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use TXFE (in OTG_DIEPINTx) before writing the data.

• **Generic periodic IN data transfers**

This section describes a typical periodic IN data transfer.

Application requirements:
1. Application requirements 1, 2, 3, and 4 of *Generic non-periodic IN data transfers on page 2701* also apply to periodic IN data transfers, except for a slight modification of requirement 2.
   - The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To
transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met:

\[
\text{transfer size}[\text{EPNUM}] = x \times \text{MPSIZ}[\text{EPNUM}] + sp
\]
(where \(x\) is an integer \(\geq 0\), and \(0 \leq sp < \text{MPSIZ}[\text{EPNUM}]\))

If \(sp > 0\), packet count[EPNUM] = \(x + 1\)
Otherwise, packet count[EPNUM] = \(x\);
MCNT[EPNUM] = packet count[EPNUM]

– The application cannot transmit a zero-length data packet at the end of a transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet:

– transfer size[EPNUM] = 0
packet count[EPNUM] = 1
MCNT[EPNUM] = packet count[EPNUM]

2. The application can only schedule data transfers one frame at a time.

– (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
– PKTCNT = MCNT (in OTG_DIEPTSIZx)
– If XFERSIZ < MCNT × MPSIZ, the last data packet of the transfer is a short packet.
– Note that: MCNT is in OTG_DIEPTSIZx, MPSIZ is in OTG_DIEPCTLx, PKTCNT is in OTG_DIEPTSIZx and XFERSIZ is in OTG_DIEPTSIZx

3. The complete data to be transmitted in the frame must be written into the transmit FIFO by the application, before the IN token is received. Even when 1 word of the data to be transmitted per frame is missing in the transmit FIFO when the IN token is received, the core behaves as when the FIFO is empty. When the transmit FIFO is empty:

– A zero data length packet would be transmitted on the USB for isochronous IN endpoints
– A NAK handshake would be transmitted on the USB for interrupt IN endpoints

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the associated transmit FIFO for the endpoint.
3. Every time the application writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data are fetched from application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO
mode) for the frame is not present in the FIFO, then the core generates an IN token received when Tx FIFO empty interrupt for the endpoint.

- A zero-length data packet is transmitted on the USB for isochronous IN endpoints
- A NAK handshake is transmitted on the USB for interrupt IN endpoints

5. The packet count for the endpoint is decremented by 1 under the following conditions:
   - For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
   - For interrupt endpoints, when an ACK handshake is transmitted
   - When the transfer size and packet count are both 0, the transfer completed interrupt for the endpoint is generated and the endpoint enable is cleared.

6. At the “Periodic frame Interval” (controlled by PFIVL in OTG_DCFG), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates an IISOIXFR interrupt in OTG_GINTSTS.

Application programming sequence:

1. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA bits.
2. Write the data to be transmitted in the next frame to the transmit FIFO.
3. Asserting the ITTXFE interrupt (in OTG_DIEPINTx) indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
4. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
5. Asserting the XFRC interrupt (in OTG_DIEPINTx) with no ITTXFE interrupt in OTG_DIEPINTx indicates the successful completion of an isochronous IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
6. Asserting the XFRC interrupt (in OTG_DIEPINTx), with or without the ITTXFE interrupt (in OTG_DIEPINTx), indicates the successful completion of an interrupt IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
7. Asserting the incomplete isochronous IN transfer (IISOIXFR) interrupt in OTG_GINTSTS with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

- Incomplete isochronous IN data transfers

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal data flow:

1. An isochronous IN transfer is treated as incomplete in one of the following conditions:
   a) The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS).
   b) The application is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects an IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx. The application can ignore this interrupt, as it
eventually results in an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS) at the end of periodic frame.

The core transmits a zero-length data packet on the USB in response to the received IN token.

2. The application must stop writing the data payload to the transmit FIFO as soon as possible.
3. The application must set the NAK bit and the disable bit for the endpoint.
4. The core disables the endpoint, clears the disable bit, and asserts the endpoint disable interrupt for the endpoint.

Application programming sequence:
1. The application can ignore the IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx on any isochronous IN endpoint, as it eventually results in an incomplete isochronous IN transfer interrupt (in OTG_GINTSTS).
2. Assertion of the incomplete isochronous IN transfer interrupt (in OTG_GINTSTS) indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.
3. The application must read the endpoint control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.
4. The application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.
5. Program the following fields in the OTG_DIEPCTLx register to disable the endpoint:
   - SNAK = 1 in OTG_DIEPCTLx
   - EPDIS = 1 in OTG_DIEPCTLx
6. The assertion of the endpoint disabled interrupt in OTG_DIEPINTx indicates that the core has disabled the endpoint.
   - At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next microframe. To flush the data, the application must use the OTG_GRSTCTL register.

**Stalling non-isochronous IN endpoints**

This section describes how the application can stall a non-isochronous endpoint.

Application programming sequence:
1. Disable the IN endpoint to be stalled. Set the STALL bit as well.
2. EPDIS = 1 in OTG_DIEPCTLx, when the endpoint is already enabled
   - STALL = 1 in OTG_DIEPCTLx
   - The STALL bit always takes precedence over the NAK bit
3. Assertion of the endpoint disabled interrupt (in OTG_DIEPINTx) indicates to the
   application that the core has disabled the specified endpoint.
4. The application must flush the non-periodic or periodic transmit FIFO, depending on
   the endpoint type. In case of a non-periodic endpoint, the application must re-enable
   the other non-periodic endpoints that do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the
   STALL bit must be cleared in OTG_DIEPCTLx.
6. If the application sets or clears a STALL bit for an endpoint due to a
   SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the
   STALL bit must be set or cleared before the application sets up the status stage
   transfer on the control endpoint.

   Special case: stalling the control OUT endpoint

   The core must stall IN/OUT tokens if, during the data stage of a control transfer, the host
   sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the
   application must enable the ITTXFE interrupt in OTG_DIEPINTx and the OTEPDIS interrupt
   in OTG_DOEPINTx during the data stage of the control transfer, after the core has
   transferred the amount of data specified in the SETUP packet. Then, when the application
   receives this interrupt, it must set the STALL bit in the corresponding endpoint control
   register, and clear this interrupt.

62.15.7 Worst case response time

When the OTG_HS controller acts as a device, there is a worst case response time for any
tokens that follow an isochronous OUT. This worst case response time depends on the AHB
clock frequency.

The core registers are in the AHB domain, and the core does not accept another token
before updating these register values. The worst case is for any token following an
isochronous OUT, because for an isochronous transaction, there is no handshake and the
next token could come sooner. This worst case value is 7 PHY clocks when the AHB clock
is the same as the PHY clock. When the AHB clock is faster, this value is smaller.

If this worst case condition occurs, the core responds to bulk/interrupt tokens with a NAK
and drops isochronous and SETUP tokens. The host interprets this as a timeout condition
for SETUP and retries the SETUP packet. For isochronous transfers, the Incomplete
isochronous IN transfer interrupt (IISOIXFR) and Incomplete isochronous OUT transfer
interrupt (IISOOXFR) inform the application that isochronous IN/OUT packets were

Choosing the value of TRDT in OTG_GUSBCFG

The value in TRDT (OTG_GUSBCFG) is the time it takes for the MAC, in terms of PHY
clocks after it has received an IN token, to get the FIFO status, and thus the first data from
the PFC block. This time involves the synchronization delay between the PHY and AHB
clocks. The worst case delay for this is when the AHB clock is the same as the PHY clock.
In this case, the delay is 5 clocks.
Once the MAC receives an IN token, this information (token received) is synchronized to the AHB clock by the PFC (the PFC runs on the AHB clock). The PFC then reads the data from the SPRAM and writes them into the dual clock source buffer. The MAC then reads the data out of the source buffer (4 deep).

If the AHB is running at a higher frequency than the PHY, the application can use a smaller value for TRDT (in OTG_GUSBCFG).

**Figure 738** has the following signals:
- `tkn_rcvd`: Token received information from MAC to PFC
- `dynced_tkn_rcvd`: Doubled sync `tkn_rcvd`, from PCLK to HCLK domain
- `spr_read`: Read to SPRAM
- `spr_addr`: Address to SPRAM
- `spr_rdata`: Read data from SPRAM
- `srcbuf_push`: Push to the source buffer
- `srcbuf_rdata`: Read data from the source buffer. Data seen by MAC

To calculate the value of TRDT, refer to **Table 500: TRDT values (HS)**.

**Figure 738. TRDT max timing case**

![TRDT max timing case diagram](image-url)
62.15.8 OTG programming model

The OTG_HS controller is an OTG device supporting HNP and SRP. When the core is connected to an “A” plug, it is referred to as an A-device. When the core is connected to a “B” plug it is referred to as a B-device. In host mode, the OTG_HS controller turns off VBUS to conserve power. SRP is a method by which the B-device signals the A-device to turn on VBUS power. A device must perform both data-line pulsing and VBUS pulsing, but a host can detect either data-line pulsing or VBUS pulsing for SRP. HNP is a method by which the B-device negotiates and switches to host role. In Negotiated mode after HNP, the B-device suspends the bus and reverts to the device role.

A-device session request protocol

The application must set the SRP-capable bit in the core USB configuration register. This enables the OTG_HS controller to detect SRP as an A-device.

Figure 739. A-device SRP

1. DRV_VBUS = VBUS drive signal to the PHY
2. VBUS_VALID = VBUS valid signal from PHY
3. A_VALID = A-peripheral VBUS level signal to PHY
4. D+ = Data plus line
5. D- = Data minus line

The following points refer and describe the signal numeration shown in the Figure 739:

1. To save power, the application suspends and turns off port power when the bus is idle by writing the port suspend and port power bits in the host port control and status register.
2. PHY indicates port power off by deasserting the VBUS_VALID signal.
3. The device must detect SE0 for at least 2 ms to start SRP when VBUS power is off.
4. To initiate SRP, the device turns on its data line pull-up resistor for 5 to 10 ms. The OTG_HS controller detects data-line pulsing.
5. The device drives VBUS above the A-device session valid (2.0 V minimum) for VBUS pulsing. The OTG_HS controller interrupts the application on detecting SRP. The session
request detected bit is set in Global interrupt status register (SRQINT set in OTG_GINTSTS).

6. The application must service the session request detected interrupt and turn on the port power bit by writing the port power bit in the host port control and status register. The PHY indicates port power-on by asserting the VBUS_VALID signal.

7. When the USB is powered, the device connects, completing the SRP process.

**B-device session request protocol**

The application must set the SRP-capable bit in the core USB configuration register. This enables the OTG_HS controller to initiate SRP as a B-device. SRP is a means by which the OTG_HS controller can request a new session from the host.

---

**Figure 740. B-device SRP**

1. VBUS_VALID = VBUS valid signal from PHY  
2. B_VALID = B-peripheral valid session to PHY  
3. DISCHRG_VBUS = discharge signal to PHY  
4. SESS_END = session end signal to PHY  
5. CHRG_VBUS = charge VBUS signal to PHY  
6. DP = Data plus line  
7. DM = Data minus line  
8. VBUS Pulsing

The following points refer and describe the signal numeration shown in the **Figure 740**:

1. To save power, the host suspends and turns off port power when the bus is idle. The OTG_HS controller sets the early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG_HS controller sets the USB suspend bit in the core interrupt register. The OTG_HS controller informs the PHY to discharge VBUS.

2. The PHY indicates the session’s end to the device. This is the initial condition for SRP. The OTG_HS controller requires 2 ms of SE0 before initiating SRP. For a USB 1.1 full-speed serial transceiver, the application must wait until VBUS discharges to 0.2 V after BSVLD (in OTG_GOTGCTL) is deasserted. This discharge
time can be obtained from the transceiver vendor and varies from one transceiver to another.

3. The OTG_HS core informs the PHY to speed up $V_{BUS}$ discharge.

4. The application initiates SRP by writing the session request bit in the OTG control and status register. The OTG_HS controller perform data-line pulsing followed by $V_{BUS}$ pulsing.

5. The host detects SRP from either the data-line or $V_{BUS}$ pulsing, and turns on $V_{BUS}$. The PHY indicates $V_{BUS}$ power-on to the device.

6. The OTG_HS controller performs $V_{BUS}$ pulsing. The host starts a new session by turning on $V_{BUS}$, indicating SRP success. The OTG_HS controller interrupts the application by setting the session request success status change bit in the OTG interrupt status register. The application reads the session request success bit in the OTG control and status register.

7. When the USB is powered, the OTG_HS controller connects, completing the SRP process.

**A-device host negotiation protocol**

HNP switches the USB host role from the A-device to the B-device. The application must set the HNP-capable bit in the core USB configuration register to enable the OTG_HS controller to perform HNP as an A-device.

![Figure 741. A-device HNP](image)

1. **DPPULLDOWN** = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY.

2. **DMPULLDOWN** = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

The following points refer and describe the signal numeration shown in the **Figure 741**:

1. The OTG_HS controller sends the B-device a SetFeature b_hnp_enable descriptor to enable HNP support. The B-device’s ACK response indicates that the B-device supports HNP. The application must set host Set HNP enable bit in the OTG control
and status register to indicate to the OTG_HS controller that the B-device supports HNP.

2. When it has finished using the bus, the application suspends by writing the port suspend bit in the host port control and status register.

3. When the B-device observes a USB suspend, it disconnects, indicating the initial condition for HNP. The B-device initiates HNP only when it must switch to the host role; otherwise, the bus continues to be suspended.

The OTG_HS controller sets the host negotiation detected interrupt in the OTG interrupt status register, indicating the start of HNP.

The OTG_HS controller deasserts the DM pull down and DM pull down in the PHY to indicate a device role. The PHY enables the OTG_DP pull-up resistor to indicate a connect for B-device.

The application must read the current mode bit in the OTG control and status register to determine device mode operation.

4. The B-device detects the connection, issues a USB reset, and enumerates the OTG_HS controller for data traffic.

5. The B-device continues the host role, initiating traffic, and suspends the bus when done.

The OTG_HS controller sets the early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG_HS controller sets the USB suspend bit in the core interrupt register.

6. In Negotiated mode, the OTG_HS controller detects the suspend, disconnects, and switches back to the host role. The OTG_HS controller asserts the DM pull down and DM pull down in the PHY to indicate its assumption of the host role.

7. The OTG_HS controller sets the connector ID status change interrupt in the OTG interrupt status register. The application must read the connector ID status in the OTG control and status register to determine the OTG_HS controller operation as an A-device. This indicates the completion of HNP to the application. The application must read the Current mode bit in the OTG control and status register to determine host mode operation.

8. The B-device connects, completing the HNP process.

**B-device host negotiation protocol**

HNP switches the USB host role from B-device to A-device. The application must set the HNP-capable bit in the core USB configuration register to enable the OTG_HS controller to perform HNP as a B-device.
1. **DPPULLDOWN** = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY.  
**DMPULLDOWN** = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

The following points refer and describe the signal numeration shown in the Figure 742:

1. The A-device sends the SetFeature b_hnp_enable descriptor to enable HNP support. The OTG_HS controller’s ACK response indicates that it supports HNP. The application must set the device HNP enable bit in the OTG control and status register to indicate HNP support. The application sets the HNP request bit in the OTG control and status register to indicate to the OTG_HS controller to initiate HNP.

2. When it has finished using the bus, the A-device suspends by writing the port suspend bit in the host port control and status register. The OTG_HS controller sets the Early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG_HS controller sets the USB suspend bit in the core interrupt register.

The OTG_HS controller disconnects and the A-device detects SE0 on the bus, indicating HNP. The OTG_HS controller asserts the DP pull down and DM pull down in the PHY to indicate its assumption of the host role.

The A-device responds by activating its OTG_DP pull-up resistor within 3 ms of detecting SE0. The OTG_HS controller detects this as a connect.

The OTG_HS controller sets the host negotiation success status change interrupt in the OTG interrupt status register, indicating the HNP status. The application must read the host negotiation success bit in the OTG control and status register to determine
host negotiation success. The application must read the current Mode bit in the core interrupt register (OTG_GINTSTS) to determine host mode operation.

3. The application sets the reset bit (PRST in OTG_HPRT) and the OTG_HS controller issues a USB reset and enumerates the A-device for data traffic.

4. The OTG_HS controller continues the host role of initiating traffic, and when done, suspends the bus by writing the port suspend bit in the host port control and status register.

5. In Negotiated mode, when the A-device detects a suspend, it disconnects and switches back to the host role. The OTG_HS controller deasserts the DP pull down and DM pull down in the PHY to indicate the assumption of the device role.

6. The application must read the current mode bit in the core interrupt (OTG_GINTSTS) register to determine the host mode operation.

7. The OTG_HS controller connects, completing the HNP process.
63 HDMI-CEC controller (CEC)

63.1 Introduction

Consumer electronics control (CEC) is part of HDMI (high-definition multimedia interface) standard as appendix supplement 1. It contains a protocol that provides high-level control functions between various audiovisual products. CEC operates at low speeds, with minimum processing and memory overhead.

The HDMI-CEC controller provides hardware support for this protocol.

63.2 HDMI-CEC controller main features

- Complies with HDMI-CEC v1.4 specification
- Independent 32 kHz CEC kernel (refer to section RCC kernel clock distribution)
- Works in Stop mode for ultra-low-power applications
- Configurable signal-free time before start of transmission
  - Automatic by hardware, according to CEC state and transmission history
  - Fixed by software (7 timing options)
- Configurable peripheral address (OAR)
- Supports Listen mode
  - Enables reception of CEC messages sent to destination address different from OAR without interfering with the CEC line
- Configurable Rx-tolerance margin
  - Standard tolerance
  - Extended tolerance
- Receive-error detection
  - Bit rising error (BRE), with optional stop of reception (BRESTP)
  - Short bit period error (SBPE)
  - Long bit period error (LBPE)
- Configurable error-bit generation
  - on BRE detection (BREGEN)
  - on LBPE detection (LBPEGEN)
  - always generated on SBPE detection
- Transmission error detection (TXERR)
- Arbitration lost detection (ARBLST)
  - with automatic transmission retry
- Transmission underrun detection (TXUDR)
- Reception overrun detection (RXOVR)
63.3 HDMI-CEC functional description

63.3.1 HDMI-CEC pin and internal signals

The CEC bus consists of a single bidirectional line that is used to transfer data in and out of the device. It is connected to a +3.3 V supply voltage via a 27 kΩ pull-up resistor. The output stage of the device must have an open-drain or open-collector to allow a wired-AND connection.

The HDMI-CEC controller manages the CEC bidirectional line as an alternate function of a standard GPIO, assuming that it is configured as alternate function open drain. The 27 kΩ pull-up must be added externally to the microcontroller.

To not interfere with the CEC bus when the application power is removed, it is mandatory to isolate the CEC pin from the bus in such conditions. This can be done by using a MOS transistor, as shown on Figure 743.

Table 504 lists the internal signals that are exchanged between the HDMI-CEC and other functional blocks (such as RCC and EXTI).

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cec_wkup</td>
<td>Digital output</td>
<td>HDMI-CEC wakeup signal</td>
</tr>
<tr>
<td>cec_it</td>
<td>Digital output</td>
<td>HDMI-CEC interrupt signal</td>
</tr>
<tr>
<td>cec_pclk</td>
<td>Digital input</td>
<td>APB clock</td>
</tr>
<tr>
<td>cec_ker_ck</td>
<td>Digital input</td>
<td>HDMI-CEC kernel clock</td>
</tr>
</tbody>
</table>

Table 503. HDMI pin

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal type</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| CEC  | Bidirectional | Two states:  
|      |             | – 1 = high impedance  
|      |             | – 0 = low impedance  
|      |             | A 27 kΩ resistor must be added externally. |
63.3.2 HDMI-CEC block diagram

![HDMI-CEC block diagram](image1)

63.3.3 Message description

All transactions on the CEC line consist of an initiator and one or more followers. The initiator is responsible for sending the message structure and the data. The follower is the recipient of any data and is responsible for setting any acknowledgment bits.

A message is conveyed in a single frame that consists of a start bit followed by a header block and optionally an opcode and a variable number of operand blocks.

All these blocks are made of an 8-bit payload - most significant bit is transmitted first - followed by an end of message (EOM) bit and an acknowledge (ACK) bit.

The EOM bit is set in the last block of a message and kept reset in all others. In case a message contains additional blocks after an EOM is indicated, those additional blocks must be ignored. The EOM bit may be set in the header block to ‘ping’ other devices, to make sure they are active.

The acknowledge bit is always set to high impedance by the initiator so that it can be driven low either by the follower that has read its own address in the header, or by the follower that needs to reject a broadcast message.

The header consists of the source logical address field, and the destination logical address field. Note that the special address 0xF is used for broadcast messages.

![Message structure](image2)
63.3.4 Bit timing

The format of the start bit is unique and identifies the start of a message. It must be validated by its low duration and its total duration.

All remaining data bits in the message, after the start bit, have consistent timing. The high-to-low transition at the end of the data bit is the start of the next data bit except for the final bit where the CEC line remains high.

63.4 Arbitration

All devices transmitting - or retransmitting - a message onto the CEC line must ensure that it has been inactive for a number of bit periods. This signal-free time is defined as the time
starting from the final bit of the previous frame and depends on the initiating device and the
current status as shown in the figure below.

**Figure 747. Signal free time**

![Figure 747](image)

Since only one initiator is allowed at any one time, an arbitration mechanism is provided to
avoid conflict when more than one initiator begins transmitting at the same time.

CEC line arbitration starts with the leading edge of the start bit and continues until the end of
the initiator address bits within the header block. During this period, the initiator must
monitor the CEC line, if whilst driving the line to high impedance it reads it back to 0.
Assuming then it has lost arbitration, it stops transmitting and becomes a follower.

**Figure 748. Arbitration phase**

![Figure 748](image)

*Figure 749* shows an example for a SFT of three nominal bit periods.

**Figure 749. SFT of three nominal bit periods**

![Figure 749](image)

A configurable time window is counted before starting the transmission.

In the SFT = 0 configuration, HDMI-CEC performs automatic SFT calculation ensuring
compliance with the HDMI-CEC standard:

- 2.5 data bit periods if the CEC is the last bus initiator with unsuccessful transmission
- 4 data bit periods if the CEC is the new bus initiator
- 6 data bit periods if the CEC is the last bus initiator with successful transmission
This is done to guarantee the maximum priority to a failed transmission and the lowest one to the last initiator that completed successfully its transmission.

Otherwise there is the possibility to configure the SFT bits to count a fixed timing value. Possible values are 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5 data bit periods.

### 63.4.1 SFT option bit

In case of SFTOPT = 0 configuration, SFT starts being counted when the start-of-transmission command is set by software (TXSOM = 1).

In case of SFTOPT = 1, SFT starts automatically being counted by the HDMI-CEC device when a bus-idle or line error condition is detected. If the SFT timer is completed at the time TXSOM command is set then transmission starts immediately without latency. If the SFT timer is still running instead, the system waits until the timer elapses before transmission can start.

In case of SFTOPT = 1 a bus-event condition starting the SFT timer is detected in the following cases:

- In case of a regular end of transmission/reception, when TXEND/RXEND bits are set at the minimum nominal data bit duration of the last bit in the message (ACK bit).
- In case of a transmission error detection, SFT timer starts when the TXERR transmission error is detected (TXERR = 1).
- In case of a missing acknowledge from the CEC follower, the SFT timer starts when the TXACKE bit is set, that is at the nominal sampling time of the ACK bit.
- In case of a transmission underrun error, the SFT timer starts when the TXUDR bit is set at the end of the ACK bit.
- In case of a receive error detection implying reception abort, the SFT timer starts at the same time the error is detected. If an error bit is generated, then SFT starts being counted at the end of the error bit.
- In case of a wrong start bit or of any uncodified low impedance bus state from idle, the SFT timer is restarted as soon as the bus comes back to hi-impedance idleness.

### 63.5 Error handling

#### 63.5.1 Bit error

If a data bit - excluding the start bit - is considered invalid, the follower is expected to notify such error by generating a low bit period on the CEC line of 1.4 to 1.6 times the nominal data bit period (3.6 ms nominally).

**Figure 750. Error bit timing**

```
ERROR BIT
  high impedance
  low impedance

3.6 ms +/-0.24 ms
```

MS31010V1
63.5.2 Message error

A message is considered lost and therefore may be retransmitted under the following conditions:

- a message is not acknowledged in a directly addressed message
- a message is negatively acknowledged in a broadcast message
- a low impedance is detected on the CEC line while it is not expected (line error)

Three kinds of error flag can be detected when the CEC interface is receiving a data bit:

63.5.3 Bit rising error (BRE)

BRE (bit rising error) is set when a bit rising edge is detected outside the windows where it is expected (see Figure 751). BRE flag also generates a CEC interrupt if the BREIE = 1.

In the case of a BRE detection, the message reception can be stopped according to the BRESTP bit value and an error bit can be generated if BREGEN bit is set.

When BRE is detected in a broadcast message with BRESTP = 1 an error bit is generated even if BREGEN = 0 to enforce initiator’s retry of the failed transmission. Error bit generation can be disabled by configuring BREGEN = 0, BRDNOGEN = 1.

63.5.4 Short bit period error (SBPE)

SBPE is set when a bit falling edge is detected earlier than expected (see Figure 751). SBPE flag also generates a CEC interrupt if the SBPEIE = 1.

An error bit is always generated on the line in case of a SBPE error detection. An error bit is not generated upon SBPE detection only when Listen mode is set (LSTN = 1) and the following conditions are met:

- A directly addressed message is received containing SBPE
- A broadcast message is received containing SBPE AND BRDNOGEN = 1

63.5.5 Long bit period error (LBPE)

LBPE is set when a bit falling edge is not detected in a valid window (see Figure 751). LBPE flag also generates a CEC interrupt if the LBPEIE = 1.

LBPE always stops the reception, an error bit is generated on the line when LBPEGEN bit is set.

When LBPE is detected in a broadcast message an error bit is generated even if LBPEGEN = 0 to enforce initiator’s retry of the failed transmission. Error bit generation can be disabled by configuring LBPEGEN = 0, BRDNOGEN = 1.

Note: The BREGEN = 1, BRESTP = 0 configuration must be avoided.
Table 505. Error handling timing parameters

<table>
<thead>
<tr>
<th>Time</th>
<th>RXTOL</th>
<th>ms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_s$</td>
<td>x</td>
<td>0</td>
<td>Bit start event.</td>
</tr>
<tr>
<td>$T_1$</td>
<td>1</td>
<td>0.3</td>
<td>The earliest time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>$T_{n1}$</td>
<td>x</td>
<td>0.6</td>
<td>The nominal time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td>$T_2$</td>
<td>0</td>
<td>0.8</td>
<td>The latest time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>$T_{ns}$</td>
<td>x</td>
<td>1.05</td>
<td>Nominal sampling time.</td>
</tr>
<tr>
<td>$T_3$</td>
<td>1</td>
<td>1.2</td>
<td>The earliest time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1.3</td>
<td></td>
</tr>
<tr>
<td>$T_{n0}$</td>
<td>x</td>
<td>1.5</td>
<td>The nominal time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td>$T_4$</td>
<td>0</td>
<td>1.7</td>
<td>The latest time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>$T_5$</td>
<td>1</td>
<td>1.85</td>
<td>The earliest time for the start of a following bit.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2.05</td>
<td></td>
</tr>
<tr>
<td>$T_{nf}$</td>
<td>x</td>
<td>2.4</td>
<td>The nominal data bit period.</td>
</tr>
<tr>
<td>$T_6$</td>
<td>0</td>
<td>2.75</td>
<td>The latest time for the start of a following bit.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2.95</td>
<td></td>
</tr>
</tbody>
</table>
63.5.6 Transmission error detection (TXERR)

The CEC initiator sets the TXERR flag if detecting low impedance on the CEC line when it is transmitting high impedance and is not expecting a follower asserted bit. TXERR flag also generates a CEC interrupt if the TXERRIE = 1.

TXERR assertion stops the message transmission. Application is in charge to retry the failed transmission up to five times.

TXERR checks are performed differently depending on the different states of the CEC line and on the RX tolerance configuration.

**Figure 752. TXERR detection**

**Table 506. TXERR timing parameters**

<table>
<thead>
<tr>
<th>Time</th>
<th>RXTOL</th>
<th>ms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_s$</td>
<td>x</td>
<td>0</td>
<td>Bit start event.</td>
</tr>
<tr>
<td>$T_1$</td>
<td>1</td>
<td>0.3</td>
<td>The earliest time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>$T_{nf}$</td>
<td>x</td>
<td>0.6</td>
<td>The nominal time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td>$T_2$</td>
<td>0</td>
<td>0.8</td>
<td>The latest time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>$T_{ns}$</td>
<td>x</td>
<td>1.05</td>
<td>Nominal sampling time.</td>
</tr>
<tr>
<td>$T_3$</td>
<td>1</td>
<td>1.2</td>
<td>The earliest time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1.3</td>
<td></td>
</tr>
</tbody>
</table>
### Table 506. TXERR timing parameters (continued)

<table>
<thead>
<tr>
<th>Time</th>
<th>RXTOL</th>
<th>ms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{n0}$</td>
<td>x</td>
<td>1.5</td>
<td>The nominal time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td>$T_4$</td>
<td>0</td>
<td>1.7</td>
<td>The latest time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>$T_5$</td>
<td>1</td>
<td>1.85</td>
<td>The earliest time for the start of a following bit.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2.05</td>
<td></td>
</tr>
<tr>
<td>$T_{nf}$</td>
<td>x</td>
<td>2.4</td>
<td>The nominal data bit period.</td>
</tr>
<tr>
<td>$T_6$</td>
<td>0</td>
<td>2.75</td>
<td>The latest time for the start of a following bit.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2.95</td>
<td></td>
</tr>
</tbody>
</table>

### 63.6 HDMI-CEC interrupts

An interrupt can be produced:
- during reception if a receive block transfer is finished or if a receive error occurs.
- during transmission if a transmit block transfer is finished or if a transmit error occurs.

### Table 507. HDMI-CEC interrupts

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rx-byte received</td>
<td>RXBR</td>
<td>RXBRIE</td>
</tr>
<tr>
<td>End of reception</td>
<td>RXEND</td>
<td>RXENDIE</td>
</tr>
<tr>
<td>Rx-overrun</td>
<td>RXOVR</td>
<td>RXOVRIE</td>
</tr>
<tr>
<td>Rxbits rising error</td>
<td>BRE</td>
<td>BREIE</td>
</tr>
<tr>
<td>Rx-short bit period error</td>
<td>SBPE</td>
<td>SBPEIE</td>
</tr>
<tr>
<td>Rx-long bit period error</td>
<td>LBPE</td>
<td>LBPEIE</td>
</tr>
<tr>
<td>Rx-missing acknowledge error</td>
<td>RXACKE</td>
<td>RXACKEIE</td>
</tr>
<tr>
<td>Arbitration lost</td>
<td>ARBLST</td>
<td>ARBLSTIE</td>
</tr>
<tr>
<td>Tx-byte request</td>
<td>TXBR</td>
<td>TXBRIE</td>
</tr>
<tr>
<td>End of transmission</td>
<td>TXEND</td>
<td>TXENDIE</td>
</tr>
<tr>
<td>Tx-buffer underrun</td>
<td>TXUDR</td>
<td>TXUDRIE</td>
</tr>
<tr>
<td>Tx-error</td>
<td>TXERR</td>
<td>TXERRIE</td>
</tr>
<tr>
<td>Tx-missing acknowledge error</td>
<td>TXACKE</td>
<td>TXACKEIE</td>
</tr>
</tbody>
</table>
63.7 HDMI-CEC registers

Refer to Section 1.2 on page 97 for a list of abbreviations used in register descriptions.

63.7.1 CEC control register (CEC_CR)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 TXEOM: Tx end of message

The TXEOM bit is set by software to command transmission of the last byte of a CEC message. TXEOM is cleared by hardware at the same time and under the same conditions as for TXSOM.

0: TXDR data byte is transmitted with EOM = 0
1: TXDR data byte is transmitted with EOM = 1

Note: TXEOM must be set when CECEN = 1.

TXEOM must be set before writing transmission data to TXDR.

If TXEOM is set when TXSOM = 0, transmitted message consists of 1 byte (HEADER) only (PING message).

Bit 1 TXSOM: Tx start of message

TXSOM is set by software to command transmission of the first byte of a CEC message. If the CEC message consists of only one byte, TXEOM must be set before of TXSOM.

Start-bit is effectively started on the CEC line after SFT is counted. If TXSOM is set while a message reception is ongoing, transmission starts after the end of reception.

TXSOM is cleared by hardware after the last byte of the message is sent with a positive acknowledge (TXEND = 1), in case of transmission underrun (TXUDR = 1), negative acknowledge (TXACKE = 1), and transmission error (TXERR = 1). It is also cleared by CECEN = 0. It is not cleared and transmission is automatically retried in case of arbitration lost (ARBLST = 1).

TXSOM can be also used as a status bit informing application whether any transmission request is pending or under execution. The application can abort a transmission request at any time by clearing the CECEN bit.

0: No CEC transmission is on-going
1: CEC transmission command

Note: TXSOM must be set when CECEN = 1.

TXSOM must be set when transmission data is available into TXDR.

HEADER first four bits containing own peripheral address are taken from TXDR[7:4], not from CEC_CFGR.OAR that is used only for reception.
**Bit 0 CECEN:** CEC enable
The CECEN bit is set and cleared by software. CECEN = 1 starts message reception and enables the TXSOM control. CECEN = 0 disables the CEC peripheral, clears all bits of CEC_CR register and aborts any on-going reception or transmission.

0: CEC peripheral is off.
1: CEC peripheral is on.

### 63.7.2 CEC configuration register (CEC_CFGR)
This register is used to configure the HDMI-CEC controller.

**Address offset:** 0x04

**Reset value:** 0x0000 0000

**Caution:** It is mandatory to write CEC_CFGR only when CECEN = 0.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
<td>LSTN</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bit 31 LSTN:** Listen mode
LSTN bit is set and cleared by software.

0: CEC peripheral receives only message addressed to its own address (OAR). Messages addressed to different destination are ignored. Broadcast messages are always received.

1: CEC peripheral receives messages addressed to its own address (OAR) with positive acknowledge. Messages addressed to different destination are received, but without interfering with the CEC bus: no acknowledge sent.

**Bits 30:16 OAR[14:0]:** Own addresses configuration
The OAR bits are set by software to select which destination logical addresses has to be considered in receive mode. Each bit, when set, enables the CEC logical address identified by the given bit position.
At the end of HEADER reception, the received destination address is compared with the enabled addresses. In case of matching address, the incoming message is acknowledged and received. In case of non-matching address, the incoming message is received only in listen mode (LSTN = 1), but without acknowledge sent. Broadcast messages are always received.

Example:
OAR = 0b000 0000 0010 0001 means that CEC acknowledges addresses 0x0 and 0x5.
Consequently, each message directed to one of these addresses is received.

**Bits 15:9** Reserved, must be kept at reset value.

**Bit 8 SFTOP:** SFT option bit
The SFTOPT bit is set and cleared by software.

0: SFT timer starts when TXSOM is set by software.
1: SFT timer starts automatically at the end of message transmission/reception.
Bit 7  **BRDNOGEN**: Avoid error-bit generation in broadcast
The BRDNOGEN bit is set and cleared by software.
0: BRE detection with BRESTP = 1 and BREGEN = 0 on a broadcast message generates an error-bit on the CEC line. LBPE detection with LBPEGEN = 0 on a broadcast message generates an error-bit on the CEC line.
1: Error-bit is not generated in the same condition as above. An error-bit is not generated even in case of an SBPE detection in a broadcast message if listen mode is set.

Bit 6  **LBPEGEN**: Generate error-bit on long bit period error
The LBPEGEN bit is set and cleared by software.
0: LBPE detection does not generate an error-bit on the CEC line.
1: LBPE detection generates an error-bit on the CEC line.

*Note:* If BRDNOGEN = 0, an error-bit is generated upon LBPE detection in broadcast even if LBPEGEN = 0.

Bit 5  **BREGEN**: Generate error-bit on bit rising error
The BREGEN bit is set and cleared by software.
0: BRE detection does not generate an error-bit on the CEC line.
1: BRE detection generates an error-bit on the CEC line (if BRESTP is set).

*Note:* If BRDNOGEN = 0, an error-bit is generated upon BRE detection with BRESTP = 1 in broadcast even if BREGEN = 0.

Bit 4  **BRESTP**: Rx-stop on bit rising error
The BRESTP bit is set and cleared by software.
0: BRE detection does not stop reception of the CEC message. Data bit is sampled at 1.05 ms.
1: BRE detection stops message reception.

Bit 3  **RXTOL**: Rx-tolerance
The RXTOL bit is set and cleared by software.
0: Standard tolerance margin:
   – Start-bit: +/- 200 µs rise, +/- 200 µs fall
   – Data-bit: +/- 200 µs rise. +/- 350 µs fall
1: Extended tolerance
   – Start-bit: +/- 400 µs rise, +/- 400 µs fall
   – Data-bit: +/- 300 µs rise, +/- 500 µs fall

Bits 2:0  **SFT[2:0]**: Signal free time
SFT bits are set by software. In the SFT = 0x0 configuration, the number of nominal data bit periods waited before transmission is ruled by hardware according to the transmission history. In all the other configurations the SFT number is determined by software.
0x0
   – 2.5 data-bit periods if CEC is the last bus initiator with unsuccessful transmission (ARBLST = 1, TXERR = 1, TXUDR = 1 or TXACKE = 1)
   – 4 data-bit periods if CEC is the new bus initiator
   – 6 data-bit periods if CEC is the last bus initiator with successful transmission (TXEOM = 1)
0x1: 0.5 nominal data bit periods
0x2: 1.5 nominal data bit periods
0x3: 2.5 nominal data bit periods
0x4: 3.5 nominal data bit periods
0x5: 4.5 nominal data bit periods
0x6: 5.5 nominal data bit periods
0x7: 6.5 nominal data bit periods
63.7.3 CEC Tx data register (CEC_TXDR)
Address offset: 0x8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.
Bits 7:0 TXD[7:0]: Tx data
TXD is a write-only register containing the data byte to be transmitted.

63.7.4 CEC Rx data register (CEC_RXDR)
Address offset: 0xC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.
Bits 7:0 RXD[7:0]: Rx data
RXD is read-only and contains the last data byte that has been received from the CEC line.

63.7.5 CEC interrupt and status register (CEC_ISR)
Address offset: 0x10
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Res, Res, Res, TX_ACKE, TX_ERR, TX_UDR, TX_END, TXBR, ARB,LST, RX_ACKE, LBPE, SBPE, BRE, RX_OVR, RX_END, RXBR
rc_w1, rc_w1, rc_w1, rc_w1, rc_w1, rc_w1, rc_w1, rc_w1, rc_w1, rc_w1, rc_w1, rc_w1, rc_w1, rc_w1, rc_w1
Bits 31:13 Reserved, must be kept at reset value.

**Bit 12 TXACKE:** Tx-missing acknowledge error

In transmission mode, TXACKE is set by hardware to inform application that no acknowledge was received. In case of broadcast transmission, TXACKE informs application that a negative acknowledge was received. TXACKE aborts message transmission and clears TXSOM and TXEOM controls.

TXACKE is cleared by software write at 1.

**Bit 11 TXERR:** Tx-error

In transmission mode, TXERR is set by hardware if the CEC initiator detects low impedance on the CEC line while it is released. TXERR aborts message transmission and clears TXSOM and TXEOM controls.

TXERR is cleared by software write at 1.

**Bit 10 TXUDR:** Tx-buffer underrun

In transmission mode, TXUDR is set by hardware if application was not in time to load TXDR before of next byte transmission. TXUDR aborts message transmission and clears TXSOM and TXEOM control bits.

TXUDR is cleared by software write at 1.

**Bit 9 TXEND:** End of transmission

TXEND is set by hardware to inform application that the last byte of the CEC message has been successfully transmitted. TXEND clears the TXSOM and TXEOM control bits.

TXEND is cleared by software write at 1.

**Bit 8 TXBR:** Tx-byte request

TXBR is set by hardware to inform application that the next transmission data has to be written to TXDR. TXBR is set when the 4th bit of currently transmitted byte is sent. Application must write the next byte to TXDR within six nominal data-bit periods before transmission underrun error occurs (TXUDR).

TXBR is cleared by software write at 1.

**Bit 7 ARBLST:** Arbitration lost

ARBLST is set by hardware to inform application that CEC device is switching to reception due to arbitration lost event following the TXSOM command. ARBLST can be due either to a contending CEC device starting earlier or starting at the same time but with higher HEADER priority. After ARBLST assertion TXSOM bit keeps pending for next transmission attempt.

ARBLST is cleared by software write at 1.

**Bit 6 RXACKE:** Rx-missing acknowledge

In receive mode, RXACKE is set by hardware to inform application that no acknowledge was seen on the CEC line. RXACKE applies only for broadcast messages and in listen mode also for not directly addressed messages (destination address not enabled in OAR). RXACKE aborts message reception.

RXACKE is cleared by software write at 1.

**Bit 5 LBPE:** Rx-long bit period error

LBPE is set by hardware in case a data-bit waveform is detected with long bit period error. LBPE is set at the end of the maximum bit-extension tolerance allowed by RXTOL, in case falling edge is still longing. LBPE always stops reception of the CEC message. LBPE generates an error-bit on the CEC line if LBPEGEN = 1. In case of broadcast, error-bit is generated even in case of LBPEGEN = 0.

LBPE is cleared by software write at 1.
Bit 4 **SBPE**: Rx-short bit period error
SBPE is set by hardware in case a data-bit waveform is detected with short bit period error. SBPE is set at the time the anticipated falling edge occurs. SBPE generates an error-bit on the CEC line. SBPE is cleared by software write at 1.

Bit 3 **BRE**: Rx-bit rising error
BRE is set by hardware in case a data-bit waveform is detected with bit rising error. BRE is set either at the time the misplaced rising edge occurs, or at the end of the maximum BRE tolerance allowed by RXTOL, in case rising edge is still longing. BRE stops message reception if BRESTP = 1. BRE generates an error-bit on the CEC line if BREGEN = 1. BRE is cleared by software write at 1.

Bit 2 **RXOVR**: Rx-overrun
RXOVR is set by hardware if RXBR is not yet cleared at the time a new byte is received on the CEC line and stored into RXD. RXOVR assertion stops message reception so that no acknowledge is sent. In case of broadcast, a negative acknowledge is sent. RXOVR is cleared by software write at 1.

Bit 1 **RXEND**: End of reception
RXEND is set by hardware to inform application that the last byte of a CEC message is received from the CEC line and stored into the RXD buffer. RXEND is set at the same time of RXBR. RXEND is cleared by software write at 1.

Bit 0 **RXBR**: Rx-byte received
The RXBR bit is set by hardware to inform application that a new byte has been received from the CEC line and stored into the RXD buffer. RXBR is cleared by software write at 1.

### 63.7.6 CEC interrupt enable register (CEC_IER)

Address offset: 0x14
Reset value: 0x0000 0000

**Caution:** It is mandatory to write CEC_IER only when CECEN = 0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **TXACKIE**: Tx-missing acknowledge error interrupt enable
The TXACKIE bit is set and cleared by software.
0: TXACKIE interrupt disabled
1: TXACKIE interrupt enabled

Bit 11 **TXERRIE**: Tx-error interrupt enable
The TXERRIE bit is set and cleared by software.
0: TXERR interrupt disabled
1: TXERR interrupt enabled
Bit 10 **TXUDRIE**: Tx-underrun interrupt enable  
The TXUDRIE bit is set and cleared by software.  
0: TXUDR interrupt disabled  
1: TXUDR interrupt enabled

Bit 9 **TXENDIE**: Tx-end of message interrupt enable  
The TXENDIE bit is set and cleared by software.  
0: TXEND interrupt disabled  
1: TXEND interrupt enabled

Bit 8 **TXBRIE**: Tx-byte request interrupt enable  
The TXBRIE bit is set and cleared by software.  
0: TXBR interrupt disabled  
1: TXBR interrupt enabled

Bit 7 **ARBLSTIE**: Arbitration lost interrupt enable  
The ARBLSTIE bit is set and cleared by software.  
0: ARBLST interrupt disabled  
1: ARBLST interrupt enabled

Bit 6 **RXACKIE**: Rx-missing acknowledge error interrupt enable  
The RXACKIE bit is set and cleared by software.  
0: RXACKE interrupt disabled  
1: RXACKE interrupt enabled

Bit 5 **LBPEIE**: Long bit period error interrupt enable  
The LBPEIE bit is set and cleared by software.  
0: LBPE interrupt disabled  
1: LBPE interrupt enabled

Bit 4 **SBPEIE**: Short bit period error interrupt enable  
The SBPEIE bit is set and cleared by software.  
0: SBPE interrupt disabled  
1: SBPE interrupt enabled

Bit 3 **BREIE**: Bit rising error interrupt enable  
The BREIE bit is set and cleared by software.  
0: BRE interrupt disabled  
1: BRE interrupt enabled

Bit 2 **RXOVRIE**: Rx-buffer overrun interrupt enable  
The RXOVRIE bit is set and cleared by software.  
0: RXOVR interrupt disabled  
1: RXOVR interrupt enabled

Bit 1 **RXENDIE**: End of reception interrupt enable  
The RXENDIE bit is set and cleared by software.  
0: RXEND interrupt disabled  
1: RXEND interrupt enabled

Bit 0 **RXBRIE**: Rx-byte received interrupt enable  
The RXBRIE bit is set and cleared by software.  
0: RXBR interrupt disabled  
1: RXBR interrupt enabled
### 63.7.7 HDMI-CEC register map

The following table summarizes the HDMI-CEC registers.

#### Table 508. HDMI-CEC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CEC_CR</td>
<td>0x04</td>
<td>CEC_CFG</td>
</tr>
<tr>
<td>0x08</td>
<td>CEC_TXDR</td>
<td>0x0C</td>
<td>CEC_RXDR</td>
</tr>
<tr>
<td>0x10</td>
<td>CEC_ISR</td>
<td>0x14</td>
<td>CEC_IER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CEC_CR</td>
<td></td>
<td>0x04</td>
<td>CEC_CFG</td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>CEC_TXDR</td>
<td></td>
<td>0x0C</td>
<td>CEC_RXDR</td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>CEC_ISR</td>
<td></td>
<td>0x14</td>
<td>CEC_IER</td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
64 Debug infrastructure

64.1 Introduction

The debug infrastructure allows software designers to debug and trace their embedded software.

The debug features can be controlled via a JTAG/serial-wire debug (SWD) access port, using industry standard debugging tools. A trace port allows data to be captured for logging and analysis.

The trace and debug system has been designed to support a variety of typical use cases:

- **Low cost trace**
  Limited trace capability is available over the single-wire debug output. This supports code instrumentation using "printf", tracing of data and address watchpoints, interrupt detection and program counter sampling. Single-wire trace can be maintained even when the processor is switched off or clock-stopped.

- **Breakpoint debugging**
  The processor can be debugged using equipment connected to the JTAG/SWD port. This allows breakpoint and watchpoint setting, code stepping, memory access.

- **Tracing code execution via the trace port**
  Trace information is combined into a single trace stream and output to a trace port analyzer in real time. An ID embedded in the trace allows the analyzer to identify the source of information packet.

- **Capturing trace continuously in a circular buffer**
  Instead of streaming it off-chip, the combined trace information can be stored on-chip in a circular buffer. The trace storage can be started and stopped by a debugger command, a software command, an external trigger signal or an internal event for example.

- **Draining the buffer to the trace port**
  The stored trace can be dumped off-chip to the trace port analyzer. The buffer draining can be initiated by the debugger, the software, an external trigger or an internal event for example.

- **Reading the buffer with the debugger**
  The debugger can read the content of the trace buffer via the debug port. This is slower than the trace port but allows basic trace functionality on the debugger without the cost of a trace port analyzer.

- **Analyzing stored trace in software**
  The trace buffer can be either read by the processor core, or transferred into the system memory by DMA. This powerful feature allows built-in test software to monitor code execution in real time, to analyze and identify faults or to autonomously handle exceptions.

- **Uploading stored trace**
  The stored trace can also be uploaded to a host machine using one of the MCU communication interfaces (such as USB, USART, SPI, I2C, Ethernet, CAN). This is especially useful if the trace port is not accessible, for example remote monitoring and failure analysis of a deployed product.
64.2 Debug infrastructure features

A comprehensive set of trace and debug features is provided to support software development and system integration:

- Breakpoint debugging
- Code execution tracing
- Software instrumentation
- Cross-triggering
- JTAG debug port
- Serial-wire debug (SWD) port
- Trigger input and output
- Serial-wire trace port
- Trace port
- Arm® CoreSight™ debug and trace components

The CoreSight components are described at high level in this document:

- SWJ-DP: JTAG/Serial-wire debug port
- AXI-AP: AXI access port
- AHB-AP: AHB access port
- APB-AP: APB access port
- ITM: instrumentation trace macrocell
- DWT: data watchpoint and trace
- ETM: embedded trace macrocell
- ETF: embedded trace FIFO
- TPIU: trace port interface unit
- SWO: serial-wire output
- CTI: cross trigger interface
- CSTF: trace bus funnel

Detailed information is available in the Arm® documents referenced in Section 64.7.
64.3 Debug infrastructure functional description

64.3.1 Debug infrastructure block diagram

The block diagram shows the logical partitioning of the debug infrastructure.

![Block diagram of debug infrastructure](image_url)

64.3.2 Debug infrastructure pins and internal signals

<table>
<thead>
<tr>
<th>Pin name</th>
<th>JTAG debug port</th>
<th>Serial-wire debug port</th>
<th>Pin assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>JTMS/SWDIO</td>
<td>I JTAG test mode select</td>
<td>IO Serial-wire data in/out</td>
<td>PA13</td>
</tr>
<tr>
<td>JTCK/SWCLK</td>
<td>I JTAG test clock</td>
<td>I Serial-wire clock</td>
<td>PA14</td>
</tr>
<tr>
<td>JTDI</td>
<td>I JTAG test data input</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>JTDI</td>
<td>I JTAG test data input</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>nJTRST</td>
<td>I JTAG test reset</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 510. Trace port pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Type</th>
<th>Description</th>
<th>Pin assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACED0</td>
<td>O</td>
<td>Trace synchronous data out 0</td>
<td></td>
</tr>
<tr>
<td>TRACED1</td>
<td>O</td>
<td>Trace synchronous data out 1</td>
<td></td>
</tr>
<tr>
<td>TRACED2</td>
<td>O</td>
<td>Trace synchronous data out 2</td>
<td></td>
</tr>
<tr>
<td>TRACED3</td>
<td>O</td>
<td>Trace synchronous data out 3</td>
<td></td>
</tr>
<tr>
<td>TRACECLK</td>
<td>O</td>
<td>Trace clock</td>
<td>Refer to datasheet</td>
</tr>
</tbody>
</table>

Table 511. Serial-wire trace port pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Type</th>
<th>Description</th>
<th>Pin assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACESWO</td>
<td>O</td>
<td>Single wire trace asynchronous data out</td>
<td>PB3(^{(1)})</td>
</tr>
</tbody>
</table>

1. TRACESWO is multiplexed with JTD0. This means that single-wire trace is only available when using the serial-wire debug interface, not when using JTAG.

Table 512. Trigger pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Type</th>
<th>Description</th>
<th>Pin assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRGIN</td>
<td>I</td>
<td>External trigger input</td>
<td>Refer to datasheet</td>
</tr>
<tr>
<td>TRGOUT</td>
<td>O</td>
<td>External trigger output</td>
<td></td>
</tr>
<tr>
<td>TRGIO</td>
<td>IO</td>
<td>External trigger bi-directional(^{(1)})</td>
<td></td>
</tr>
</tbody>
</table>

1. TRGIO can be configured as an input or an output by the TRGOEN bit in the MCU debug unit (DBGMCU). If configured as an input, it is connected to TRGIN. If configured as an output, it is connected to TRGOUT. This is because TRGIN and TRGOUT are not available on certain packages.
64.3.3 Debug infrastructure powering, clocking and reset

Power domains

The debug components are distributed across the power CPU domain (CD) and SmartRun domain (SRD). The SmartRun power domain must be always on when the debugger is connected. It therefore contains the SWJ-DP, so that the debugger does not lose the connection with the SoC when the CPU power domain is switched off. In addition, it contains the MCU debug unit (DBGMCU) and the serial-wire trace features.

The CPU power domain contains the Cortex-M7 core and the associated debug and trace components. It also contains the system trace components located on the APB-D. This power domain therefore needs to be on whenever a debug access to the Cortex-M7 is required or whenever a trace functionality is active on the processor.
The debugger supplies the clock for the debug port, SWTCLK, via the debug interface pin JTCK/SWCLK. This clock is used to register the serial input data in both Serial-wire and JTAG modes, as well as to operate the state machines and internal logic of the debug port. It must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the DAPCLK domain, which covers the rest of the SWJ-DP and the access ports. The DBGMCU and a ROM table are also in the DAPCLK domain.

CK_DBG_SRD clocks the SWO.

Both DAPCLK and CK_DBG_SRD are gated versions of the SmartRun domain system clock (rcc_fclk_srd).

CK_DBG_CD clocks the trace components in the CPU power domain: a ROM table, CoreSight trace funnel, ETF, system CTI and TPIU. It is a gated version of the CPU domain system clock (rcc_aclk).

TRACECLK is the trace port output clock. It is a gated version of the CPU domain system clock (rcc_aclk).

All the debug clocks (except DAPCLK) can be enabled and disabled by register bits in the DBGMCU. The DAPCLK domain is enabled by the debugger using the CDBGPWRUPREQ bit in the debug port CTRL/STAT register. The clock must be enabled before the debugger can access any of the debug features on the device. It must be disabled at power up and when the debugger is disconnected, to avoid wasting energy.
The debug and trace components included in the processor (such as ETM, ITM, DWG, FPB) are clocked with the CPU clock (rcc_cpu_ck).

**Debug with low-power modes**

The device includes power-saving features allowing individual power domains to be switched off or stopped when not required. If a power domain is switched off or not clocked, all debug components in that domain are inaccessible to the debugger. To avoid this, power saving mode emulation is implemented. If the emulation is enabled for a domain, the domain still enters power saving mode but its clock and power are maintained. In other words, the domain behaves as if it is in power saving mode while the debugger does not lose the connection.

The emulation mode is programmed in the DBGMCU. For more information, refer to Section 64.5.7.

**Reset of debug infrastructure**

The debug components, except for the debug and access ports, are reset by their respective power domain resets. The debug port (SWJ-DP) is reset by a power-on reset of the SmartRun domain only.

### 64.4 Debug access port functional description

The debug access port (DAP) is a debug subsystem comprising Serial-wire and JTAG debug port (SWJ-DP) and three access ports.

#### 64.4.1 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight component that implements an external access port for connecting debugging equipment.

The port can be configured as:

- a 5-pin standard JTAG debug port (JTAG-DP)
- a 2-pin (clock + data) Serial-wire debug port (SW-DP)

The two modes are mutually exclusive, since they share the same I/O pins.

By default, the JTAG-DP is selected upon a system or power-on reset. The five I/Os are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on the JTDI, JTMS/SWDIO and nJTRST lines, as well as a pull-down resistor on the JTCK/SWCLK line.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

```
...(50 or more),....0,1,1,1,0,0,1,1,1,0,0,1,1,1,0,0,1,1,1,...,(50 or more),...
```

JTCK/SWCLK must be cycled for each data bit.

In SW-DP mode, the unused JTAG lines JTDI, JTDO and nJTRST can be used for other functions.

All SWJ port I/Os can be reconfigured to other functions by software. In that case, debugging is no longer possible.
Serial-wire debug port

The Serial-wire debug protocol uses two pins:
- SWCLK: clock from host to target
- SWDIO: bi-directional serial data (100 kΩ pull-up required)

Serial data is transferred LSB first, synchronously with the clock. A transfer comprises three phases:
1. packet request (8 bits) transmitted by the host
2. acknowledge response (3 bits) transmitted by the target
3. data transfer (33 bits) transmitted by the host (in case of a write) or target
   (in case of a read)

The data transfer only occurs if the acknowledge response is OK.
Between each phase, if the direction of the data is reversed, a single clock cycle turn-around time is inserted.

### Table 513. Packet request

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Start</td>
<td>Must be 1</td>
</tr>
<tr>
<td>1</td>
<td>APnDP</td>
<td>0: DP register access - see Table 517 for a list of DP registers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: AP register access - see Section 64.4.2</td>
</tr>
<tr>
<td>2</td>
<td>RnW</td>
<td>0: Write request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Read request</td>
</tr>
<tr>
<td>4:3</td>
<td>A(3:2)</td>
<td>Address field of the DP or AP register (refer to Table 517 and Table 518)</td>
</tr>
<tr>
<td>5</td>
<td>Parity</td>
<td>Single bit parity of preceding bits</td>
</tr>
<tr>
<td>6</td>
<td>Stop</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Park</td>
<td>Not driven by host. Must be read as 1 by target.</td>
</tr>
</tbody>
</table>

### Table 514. ACK response

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:0</td>
<td>ACK</td>
<td>000b: FAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010b: WAIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100b: OK</td>
</tr>
</tbody>
</table>

### Table 515. Data transfer

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>WDATA or RDATA</td>
<td>Write or Read data</td>
</tr>
<tr>
<td>32</td>
<td>Parity</td>
<td>Single bit parity of 32 data bits</td>
</tr>
</tbody>
</table>
Figure 756 shows successful write and read transfers.

**Figure 756. SWD successful data transfer**

In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled. In this case the data are ignored by the target (in case of a write), or not driven (in case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists of 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the Serial-wire debug protocol, refer to the Arm® Debug Interface Architecture Specification [1].

**Note:** The SWJ-DP implements SWD protocol version 2.
The JTAG-DP implements a TAP state machine (TAPSM) based on IEEE 1149.1-1990. The state machine is shown in Figure 757. It controls two scan chains, one associated with an instruction register (IR) and one with a number of data registers (DR).

When the TAPSM goes through the Capture-IR state, 0b0001 is transferred onto the instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.

While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:

- The LSB of the IR scan chain is output on JTDO.
- Bit [n] of the IR scan chain is transferred to bit [n-1].
- The value on JTDI is transferred to the MSB of the IR scan chain.

When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred into the instruction register.

When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers onto one of the DR scan chains, connected between JTDI and JTDO.
The value held in the instruction register determines which data register and associated DR scan chain are selected.

This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shifts in the Shift-IR state.

When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred into the selected data register.

When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in IR.

When active, the nJTRST signal resets the state machine asynchronously to the Test-Logic-Reset state.

The data registers corresponding to the 4-bit IR instructions are listed in Table 516.

### Table 516. JTAG-DP data registers

<table>
<thead>
<tr>
<th>Instruction register</th>
<th>Data register</th>
<th>Scan chain length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 to 0111</td>
<td>(BYPASS)</td>
<td>1</td>
<td>Reserved: BYPASS selected</td>
</tr>
<tr>
<td>1000</td>
<td>ABORT</td>
<td>35</td>
<td>Abort register&lt;br&gt;– Bits 31:1 = reserved&lt;br&gt;– Bit 0 = APABORT: write 1 to generate an AP abort</td>
</tr>
<tr>
<td>1001</td>
<td>(BYPASS)</td>
<td>1</td>
<td>Reserved: BYPASS selected</td>
</tr>
<tr>
<td>1010</td>
<td>DPACC</td>
<td>35</td>
<td>Debug port access register&lt;br&gt;Initiates the debug port and allows access to a debug port register.&lt;br&gt;– When transferring data IN:&lt;br&gt;  Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request&lt;br&gt;  Bits 2:1 = A[3:2] = 2-bit address of a debug port register&lt;br&gt;  Bit 0 = RnW = Read request (1) or write request (0)&lt;br&gt;– When transferring data OUT:&lt;br&gt;  Bits 34:3 = DATA[31:0] = 32-bit data read following a read request&lt;br&gt;  Bits 2:0 = ACK[2:0] = 3-bit Acknowledge&lt;br&gt;  010 = OK/FAULT&lt;br&gt;  001 = WAIT&lt;br&gt;  Other = reserved</td>
</tr>
<tr>
<td>1011</td>
<td>APACC</td>
<td>35</td>
<td>Access port access register&lt;br&gt;Initiates an access port and allows access to an access port register.&lt;br&gt;– When transferring data IN:&lt;br&gt;  Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request&lt;br&gt;  Bits 2:1 = A[3:2] = 2-bit sub-address of an access port register&lt;br&gt;  Bit 0 = RnW = Read request (1) or write request (0)&lt;br&gt;– When transferring data OUT:&lt;br&gt;  Bits 34:3 = DATA[31:0] = 32-bit data read following a read request&lt;br&gt;  Bits 2:0 = ACK[2:0] = 3-bit Acknowledge:&lt;br&gt;  010 = OK/FAULT&lt;br&gt;  001 = WAIT&lt;br&gt;  Other = reserved</td>
</tr>
<tr>
<td>1100</td>
<td>(BYPASS)</td>
<td>1</td>
<td>Reserved: BYPASS selected</td>
</tr>
</tbody>
</table>
The DR registers are described in more detail in the Arm® Debug Interface Architecture Specification [1].

### Debug port registers

The SW-DP and JTAG-DP both access the debug port (DP) registers. These are listed in Table 517.

The debugger can access the DP registers as follows:

1. Program the SELECT register DPBANKSEL field in the DP to select the register bank to be accessed (see Table 517).
2. Program the A[3:2] field in the DPACC register, if using JTAG, with the register address within the bank. Program the R/W bit to select a read or a write. In case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and R/W fields are part of the packet request word sent to the SW-DP with the APnDP bit reset (see Table 513). The write data is sent in the data phase.

<table>
<thead>
<tr>
<th>Address</th>
<th>A[3:2] field value</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>00</td>
<td>R</td>
<td>DP_DPIDR register. It contains the IDCODE for the debug port.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W</td>
<td>DP_ABORT register(1). It aborts the current AP transaction. This register is also used to clear the error flags in the DP_CTRL/STAT register.</td>
</tr>
<tr>
<td>0x4</td>
<td>01</td>
<td>R/W</td>
<td>If DPBANKSEL[3:0] = 0x0 (DP_SELECT register):</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTRL/STAT register. It controls the DP and provides status information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If DPBANKSEL[3:0] = 0x1 (DP_SELECT register):</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DP_DLCR register(2). It controls the operating mode of the SWD data link.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If DPBANKSEL[3:0] = 0x2 (DP_SELECT register):</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DP_TARGETID register. It provides target identification information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If DPBANKSEL[3:0] = 0x3 (DP_SELECT register):</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DLPIDR register(2). It provides the SWD protocol version.</td>
</tr>
<tr>
<td>0x8</td>
<td>10</td>
<td>R</td>
<td>RESEND register(2). It returns the value that was returned by the last AP read or DP_RDBUFF read, used in the event of a corrupted read transfer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W</td>
<td>DP_SELECT register. It selects the access port, access port register bank, and DP register at address 0x4.</td>
</tr>
</tbody>
</table>
Debug infrastructure

Table 517. Debug port registers (continued)

<table>
<thead>
<tr>
<th>Address</th>
<th>A[3:2] field value</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0xC     | 11                 | R   | DP_RDBUFF register  
Via JTAG-DP, it is used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation).  
Via SW-DP, it contains the result of the preceding AP read access, allowing a new AP access to be avoided. |

<table>
<thead>
<tr>
<th>Address</th>
<th>A[3:2] field value</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0xC     | 11                 | W   | DP_TARGETSEL register(2). On a write to DP_TARGETSEL immediately following a line reset sequence, the target is selected if the following conditions are both met:  
Writing any other value deselects the target. Debug tools must write 0xFFFFFFFF to deselect all targets. This is an invalid DP_TARGETID value. All other invalid DP_TARGETID values are reserved. |

1. Access to the AP ABORT register from the JTAG-DP is done using the ABORT instruction.  
2. Only accessible via SW-DP. Register is “reserved” via JTAG-DP.

Debug port identification register (DP_DPIDR)

Address offset: 0x0  
Reset value: 0x6BA0 2477

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:28 **REVISION[3:0]**: revision code  
0x6

Bits 27:20 **PARTNO[7:0]**: part number for the debug port  
0xBA

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **MIN**: minimal debug port (MINDP) implementation  
0: MINDP not implemented (transaction counter and pushed operations are supported)

Bits 15:12 **VERSION[3:0]**: DP architecture version  
0x2: DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code  
0x23B: Arm®

Bit 0 Reserved, must be kept at reset value.
**Debug port abort register (DP_ABORT)**

Address offset: 0x0

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:5  Reserved, must be kept at reset value.

**Bit 4 ORUNERRCLR:** overrun error clear bit
0: no effect
1: STICKYORUN bit cleared in CTRL/STAT register

**Bit 3 WDERRCLR:** write data error clear bit
0: no effect
1: WDATAERR bit cleared in CTRL/STAT register

**Bit 2 STKERRCLR:** sticky error clear bit
0: no effect
1: STICKYERR bit cleared in CTRL/STAT register

**Bit 1 STKCMPCLR:** sticky compare clear bit
0: no effect
1: STICKYCMP bit cleared in CTRL/STAT register

**Bit 0 DAPABORT:** abort current AP transaction
The transaction is aborted if an excessive number of WAIT responses are returned, indicating that the transaction has stalled.
0: no effect
1: transaction aborted
Debug port control/status register (DP_CTRL/STAT)

Address offset: 0x4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>CSYSPWRUPACK</th>
<th>system domain power-up status bit - not used in this device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 30</td>
<td>CSYSPWRUPREQ</td>
<td>system domain power-up control bit - not used in this device</td>
</tr>
<tr>
<td>Bit 29</td>
<td>CDBGWRUPACK</td>
<td>debug domain power-up status bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit is read-only. It returns the status of the debug domain power-up acknowledge signal from the power controller.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: domain powered down</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: domain powered up</td>
</tr>
<tr>
<td>Bit 28</td>
<td>CDBGWRUPREQ</td>
<td>debug domain power-up/down control bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit controls the debug domain power-up/down request signal to the power controller.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: power-down requested</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: power-up requested</td>
</tr>
<tr>
<td>Bit 27</td>
<td>CDBGRSSTACK</td>
<td>debug domain reset status bit - not used in this device</td>
</tr>
<tr>
<td>Bit 26</td>
<td>CDBGRSREQ</td>
<td>debug domain reset control bit - not used in this device</td>
</tr>
<tr>
<td>Bits 25:24</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

| Bits 23:12 | TRNCTN[11:0] | transaction counter |
|            |              | To program a sequence of transactions to incremental addresses via an AP, TRNCNT bits are loaded with the number of transactions to perform. It is decremented at the successful completion of each transaction. |

| Bits 11:8 | MASKLANE[3:0] | pushed-compare and pushed-verify masking bits |
|          |              | The field indicates the bytes to be masked in pushed-compare and pushed-verify operations (DP_CTRL/STAT register’s field TRNMODE = 1 or 2). In the pushed operations, the word supplied in an AP write transaction is compared with the current value at the target AP address. |
|          |              | 0b1XXX: include byte lane 3 in comparisons |
|          |              | 0bX1XX: include byte lane 2 in comparisons |
|          |              | 0bXX1X: include byte lane 1 in comparisons |
|          |              | 0bXXXX: include byte lane 0 in comparisons |
Bit 7 **WDATAERR**: write data error in SW-DP

The bit indicates that:

- There is a parity or a framing error on the data phase of a write operation
- A write operation that has been accepted by the DP, has then be discarded without being submitted to the AP.

This bit is read-only. It is reset by writing 1 to the WDERRCLR bit of the DP_ABORT register.

0: no error
1: an error occurred

This bit is reserved in JTAG-DP.

Bit 6 **READOK**: AP read response in SW-DP

This bit indicates the response to the last AP read access. It is read-only.

0: Read not OK
1: Read OK

This bit is reserved in JTAG-DP.

Bit 5 **STICKYERR**: transaction error (read-only in SW-DP, R/W in JTAG-DP)

This bit indicates that an error occurred during an AP transaction.

0: no error
1: an error occurred

In the SW-DP, this bit is reset by writing 1 to the STKERRCLR bit of the DP_ABORT register.
In the JTAG-DP, this bit is reset by programming it to 1.

Bit 4 **STICKYCMP**: compare match (read-only in SW-DP, R/W in JTAG-DP)

This bit indicates that a match occurred in a pushed operation.

0: match if TRNMODE = 0x1; no match if TRNMODE = 0x2
1: no match if TRNMODE = 0x1; match if TRNMODE = 0x2

In the SW-DP, this bit is reset by writing 1 to the STKCMPCLR bit in the DP_ABORT register.
In the JTAG-DP, this bit is reset by programming it to 1.
Bits 3:2  **TRNMODE[1:0]**: transfer mode for AP write operations

For read operations, this field must be set to 0x0.
0x0: normal operation. AP transactions are passed directly to the AP.
0x1: pushed-verify operation.
The DP stores the write data and performs a read transaction at the target AP address. The result of the read operation is compared with the stored data. If they do not match, the STICKYCMP bit is set.
0x2: pushed-compare operation.
The DP stores the write data and performs a read transaction at the target AP address. The result of the read is compared with the stored data. If they match, the STICKYCMP bit is set.
0x3: reserved
In pushed operations, only the data bytes indicated by the MASKLANE field are included in the comparison.

Bit 1  **STICKYORUN**: overrun (read-only in SW-DP, R/W in JTAG-DP)

This bit indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set.
0: no overrun
1: an overrun occurred
In the SW-DP, this bit is reset by writing 1 to the ABORT register’s ORUNERRCLR bit. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bit 0  **ORUNDETECT**: overrun detection mode enable

0: overrun detection disabled
1: overrun detection enabled.
In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.

**Debug port data link control register (DP_DLCR)**

Address offset: 0x4
Reset value: 0x0000 0040

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RM0455 Debug infrastructure

Bits 31:10  Reserved, must be kept at reset value.

Bits 9:8  **TURNROUND[1:0]**: tristate period for SWDIO
0x0: 1 data bit period
0x1: 2 data bit periods
0x2: 3 data bit periods
0x3: 4 data bit periods

Bits 7:0  Reserved, must be kept at reset value.

**Debug port target identification register (DP_TARGETID)**

Address offset: 0x4
Reset value: 0x1048 0401

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREV</td>
<td>ESION[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>R</td>
<td>E</td>
<td>V</td>
<td>I</td>
<td>S</td>
<td>I</td>
<td>O</td>
<td>[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPARTNO[15:4]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP ART</td>
<td>N</td>
<td>O</td>
<td>[15:4]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28  **TREV\SION[3:0]**: target revision
0x1: revision 1

Bits 27:12  **TPARTNO[15:0]**: target part number
0x0480: STM32H7

Bits 11:1  **TDESIGNER[10:0]**: target designer JEDEC code
0x020: STMicroelectronics

Bit 0  Reserved, must be kept at reset value.

**Debug port data link protocol identification register (DP_DLPIDR)**

Address offset: 0x4
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TINSTANCE[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TINST</td>
<td>A</td>
<td>N</td>
<td>S</td>
<td>T</td>
<td>I</td>
<td>C</td>
<td>E</td>
<td>N</td>
<td>[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROTSVN[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRO T</td>
<td>S</td>
<td>V</td>
<td>N</td>
<td>[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
Debug infrastructure

**Debug port resend register (DP_RESEND)**

Address offset: 0x8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESEND[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **RESEND[31:0]**: last AP read or DP_RDBUFF read value
These bits contain the value that was returned by the last AP read or DP_RDBUFF read. Used in the event of a corrupted read transfer.

**Debug port access port select register (DP_SELECT)**

Address offset: 0x8
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>APSEL[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:28 **APSEL[3:0]**: access port select bits
These bits select the access port for the next transaction.
0x0: AP0 - Cortex-M7 debug access port (AHB-AP)
0x1: AP1 - SRD access port (AHB-AP)
0x2: AP2 - system debug access port (APB-AP)
0x3 to 0xF: reserved

Bits 31:28 **TINSTANCE[3:0]**: target instance number
These bits define the instance number for this device in a multi-drop system.
0x0: instance 0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]**: Serial-wire debug protocol version
0x1: Version 2
Bits 27:8  Reserved, must be kept at reset value.

Bits 7:4  **APBANKSEL[3:0]**: AP register bank select bits
These bits select the 4-word register bank on the active AP for the next transaction.

Bits 3:0  **DPBANKSEL[3:0]**: DP register bank select bits
These bits select the register at address 0x4 of the debug port.
0x0: CTRL/STAT register
0x1: DLCR register
0x2: TARGETID register
0x3: DLPIIDR register
0x4 to 0xF: reserved

### Debug port read buffer register (DP_RDBUFF)

Address offset: 0xC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**RDBUFF[31:16]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**RDBUFF[15:0]**

Bits 31:0  **RDBUFF[31:0]**: last AP read value
The field contains the value returned by the last AP read access. There are two ways to retrieve the value returned by an AP read access:
- Perform a second read access to the same address, it initiates a new transaction on the corresponding bus.
- Read the value returned by the last AP read access from the DP_RDBUFF register, in which case no new AP transaction occurs

### Debug port target identification register (DP_TARGETSEL)

Address offset: 0xC
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**TINSTANCE[3:0]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**TPARTNO[3:0]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
64.4.2 Access ports

The following access ports (AP) are attached to the DP:

1. **AP0**: Cortex-M7 access port (AHB-AP). Allows access to the debug and trace features integrated in the Cortex-M7 processor core via an AHB-Lite bus connected to the AHBD port of the processor.

2. **AP1**: SRD access port (AHB-AP). Allows access to the bus matrix in the SmartRun domain. This gives visibility of the SmartRun domain memory and peripherals when the CPU domain is switched off. No CoreSight components are accessible via this port.

3. **AP2**: System access port (APB-AP). Allows access to the debug and trace features on the system APB debug bus, that is, all components not included in the processor core.

All access ports are of MEM-AP type, that is, the debug and trace component registers, are mapped in the address space of the associated debug bus. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in *Table 518*.
The address of the AP registers is composed as follows:

- bits [7:4]: content of the APBANKSEL field in the DP_SELECT register
- bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see Table 516) or content of the SW-DP packet request (see Table 513), depending on the debug interface used
- bits [1:0]: always set to 0

The content of the APSEL field in DP_SELECT register defines which MEM-AP is being accessed.

The debugger can access the AP registers as follows:

1. Program the APSEL field of the DP_SELECT register to choose one of the APs, and program the APBANKSEL field to select the register bank to be accessed.
2. Program the A(3:2) field in the APACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or a write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit set (see Table 513). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the MEM-AP registers (using the above AP register access procedure) as follows:

1. Program the transaction target address in the TAR register.
2. Program the CSW register, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the DRW register to initiate a bus transaction at the address held in the TAR register. Alternatively, a read or write to banked data register BDn triggers an access to address TAR[31:4] + n (this allows accessing up to four consecutive addresses without changing the address in the TAR register).

For more detailed information on the MEM-AP, refer to the Arm® Debug Interface Architecture Specification [1].

### MEM-AP registers

#### Table 518. MEM-AP registers

<table>
<thead>
<tr>
<th>Address</th>
<th>APBANKSEL</th>
<th>A(3:2)</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0x0</td>
<td>0</td>
<td>AP_CSW</td>
<td>Control/status word register</td>
</tr>
<tr>
<td>0x04</td>
<td>0x0</td>
<td>1</td>
<td>AP_TAR</td>
<td>Transfer address register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Target address for the bus transaction.</td>
</tr>
<tr>
<td>0x08</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>0x0</td>
<td>3</td>
<td>AP_DRW</td>
<td>Data read/write register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:0].</td>
</tr>
<tr>
<td>0x10</td>
<td>0x1</td>
<td>0</td>
<td>AP_BD0</td>
<td>Banked data 0 register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] &amp;&lt;&lt; 4 + 0x0.</td>
</tr>
<tr>
<td>0x14</td>
<td>0x1</td>
<td>1</td>
<td>AP_BD1</td>
<td>Banked data 1 register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] &amp;&lt;&lt; 4 + 0x4.</td>
</tr>
</tbody>
</table>
**Access port control/status word register (AP_CSW)**

Address offset: 0x0

Reset value: 0x0000 0002 (APB-AP), 0x4000 0002 (AHB-AP)

<table>
<thead>
<tr>
<th>Address</th>
<th>APBANKSEL</th>
<th>A(3:2)</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x18</td>
<td>0x1</td>
<td>2</td>
<td>AP_BD2</td>
<td>Banked data 2 register.Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] &lt;&lt; 4 + 0x8.</td>
</tr>
<tr>
<td>0x1C</td>
<td>0x1</td>
<td>3</td>
<td>AP_BD3</td>
<td>Banked data 3 register.Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] &lt;&lt; 4 + 0xC.</td>
</tr>
<tr>
<td>0x20-0xEC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xF0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xF4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xF8</td>
<td>0xF</td>
<td>2</td>
<td>AP_BASE</td>
<td>Banked data 3 register.</td>
</tr>
<tr>
<td>0xFC</td>
<td>0xF</td>
<td>3</td>
<td>AP_IDR</td>
<td>Identification register (RO)</td>
</tr>
</tbody>
</table>

**Bit 31** Reserved, must be kept at reset value.

**Bit 30** SPROT: secure transfer request bit

In the APB-AP, this field is reserved. In the AHB-APs, this field sets the protection attribute HPROT[6] of the bus transfer.

0: If SPIDEN is high, secure transfer., if SPIDEN is low, non-secure transfer
1: non-secure transfer

**Bit 29** Reserved, must be kept at reset value.
Bits 28:24 **PROT[4:0]**: bus transfer protection bits
In the APB-AP, this field is reserved. In the AHB-APs, this field sets the protection attributes HPROT[4:0] of the bus transfer.
0bXXXX0: instruction fetch
0bXXXX1: data access
0bXXX0X: user mode
0bXXX1X: privileged mode
0bXX0XX: non-bufferable
0bXX1XX: bufferable
0bX0XXX: non-cacheable
0bX1XXX: cacheable
0b0XXXX: non-exclusive
0b1XXXX: exclusive

Bit 23 **SPISTATUS**: status of SPIDEN option bit
This bit determines whether the debugger can access secure memory. This field is reserved in the APB-AP.
0: secure AHB transfers blocked
1: secure AHB transfers allowed

Bits 22:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: barrier support enabled bits
These bits define if the memory barrier operation is supported.
0x0: not supported

Bit 7 **TRINPROG**: transfer in progress
This bit indicates that a bus transfer is in progress on the AP.
0: no transfer in progress
1: bus transfer in progress

Bit 6 **DEVICEEN**: device enable bit
This bit defines whether the AP can be accessed or not.
1: AP access enabled.

Bits 5:4 **ADDRINC[1:0]**: auto-increment mode bits
These bits define whether the TAR address is automatically incremented after a transaction.
0x0: no auto-increment
0x1: address incremented by the size in bytes of the transaction (SIZE field)
0x2: packed transfers enabled (only in AHB-APs - reserved in APB-AP). A 32-bit AP access gives rise to 1 x 32-bit, 2 x 16-bit or 4 x 8-bit bus transactions, corresponding to the programmed transaction size. The data are packed or unpacked accordingly.
0x3: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: size of next memory access transaction (only for AHB-APs)
0x0: byte (8-bit)
0x1: half-word (16-bit)
0x2: word (32-bit)
0x3-0x7: reserved
For APB-AP, this field is read-only and fixed at 0x2 (32-bit).
### Access port base address register (AP_BASE)

Address offset: 0xF8

Reset value: 0xE00F E003 (AP0), 0x0000 0002 (AP1), 0xE00E 0003 (AP2)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:12 **BASEADDR[19:0]**: base address (bits 31 to 12) of ROM table for the AP
- The 12 LSBs are zero since the ROM table must be aligned on a 4-Kbyte boundary.
- AP0 (Cortex-M7 AHB-AP): 0xE00FE
- AP1 (SRD AHB-AP): 0x00000 (no ROM table present)
- AP2 (System APB-AP): 0xE00E0

Bits 11:2: Reserved, must be kept at reset value.

Bit 1 **FORMAT**: base address register format
- 1: Arm® debug interface v5

Bit 0 **ENTRYPRESEN T**: debug component present status bit
- This bit indicates that debug components are present on the access port bus.
- 0: debug components not present (AP1)
- 1: debug components present (AP0, AP2)

### Access port identification register (AP_IDR)

Address offset: 0xFC

Reset value: 0x6477 0001 (AP0 and AP1), 0x4477 0002 (AP2)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:24 **REVISION[3:0]**, **JEDECBank[3:0]**, **JEDECCode[6:0]**

### Memory map

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 15:2 **REVISION[3:0]**, **JEDECBank[3:0]**, **JEDECCode[6:0]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**IDENTITY[7:0]**
The trace and debug subsystem features the following CoreSight components:

- System ROM tables
- System cross-trigger interface (CTI)
- Cross-trigger matrix (CTM)
- Trace port interface unit (TPIU)
- Trace bus funnel (CSTF)
- Embedded trace FIFO (ETF)
- Serial-wire output (SWO)

These components are accessible by the debugger via the system APB-AP and its associated APB-D debug bus. They are also accessible by the Cortex-M7 processor.

The MCU debug unit (DBGMCU) is also accessed via the APB-D. This non-CoreSight component contains registers for configuring the behavior of the device in debug mode.

A trace bus replicator branches the trace bus from the CPU's ITM CoreSight component to ETF and SWO.

### 64.5.1 System ROM tables

There are two ROM tables on the APB-D bus. The ROM table is a CoreSight component that contains the base addresses of all the CoreSight components on the APB-D bus. These tables allow a debugger to discover the topology of the CoreSight components automatically.

The first table points to the second table and to the CoreSight components located in the SmartRun power domain: SWO. The DBGMCU is not referenced by the table as it is not a standard CoreSight component. The table occupies a 4-Kbyte, 32-bit wide chunk of APB-D address space, from 0xE00E0000 to 0xE00E0FFC when accessed by the debugger, and from 0x59000000 to 0x59000FFC when accessed from the system bus.
The second table occupies a 4-Kbyte, 32-bit wide chunk of APB-D address space, from 0xE00F0000 to 0xE00F0FFC when accessed by the debugger, and from 0x59010000 to 0x59010FFC when accessed from the system bus.

The top of each ROM table contains a number of read-only registers, including the standard CoreSight component and peripheral identity registers, see section System ROM registers.

Each debug component occupies one or more 4-Kbyte blocks of address space. This block of address space is referred to as the debug register file for the component.

The component address offset field of a ROM Table entry points to the start of the last 4-Kbyte block of the address space of the component. This block always contains the component and peripheral ID registers for the component, starting at offset 0xFD0 from the start of the block. The 4-Kbyte count field PIDR4 [7:4] specifies the number of 4-Kbyte blocks for the component.

**Table 519. System ROM table 1**

<table>
<thead>
<tr>
<th>Address offset in ROM table</th>
<th>Component name</th>
<th>Component base address (debugger)</th>
<th>Component base address (system bus)</th>
<th>Component address offset</th>
<th>Size</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>System ROM table 2</td>
<td>0xE00F0000</td>
<td>0x59010000</td>
<td>0x1000</td>
<td>4 Kbyte</td>
<td>0x00010003</td>
</tr>
<tr>
<td>0x004</td>
<td>SWO</td>
<td>0xE00E2000</td>
<td>0x59002000</td>
<td>0x02000</td>
<td>4 Kbyte</td>
<td>0x00002003</td>
</tr>
<tr>
<td>0x008</td>
<td>Top of table</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x00C to 0xFC8</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x0FC to 0xFFC</td>
<td>ROM table registers</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>See System ROM registers</td>
</tr>
</tbody>
</table>

**Table 520. System ROM table 2**

<table>
<thead>
<tr>
<th>Address offset in ROM table</th>
<th>Component name</th>
<th>Component base address (debugger)</th>
<th>Component base address (system bus)</th>
<th>Component address offset</th>
<th>Size</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>System CTI</td>
<td>0xE00F1000</td>
<td>0x59011000</td>
<td>0x1000</td>
<td>4 Kbyte</td>
<td>0x00001003</td>
</tr>
<tr>
<td>0x004</td>
<td>CSTF</td>
<td>0xE00F3000</td>
<td>0x59013000</td>
<td>0x3000</td>
<td>4 Kbyte</td>
<td>0x00003003</td>
</tr>
<tr>
<td>0x008</td>
<td>ETF</td>
<td>0xE00F4000</td>
<td>0x59014000</td>
<td>0x4000</td>
<td>4 Kbyte</td>
<td>0x00004003</td>
</tr>
<tr>
<td>0x00C</td>
<td>TPIU</td>
<td>0xE00F5000</td>
<td>0x59015000</td>
<td>0x5000</td>
<td>4 Kbyte</td>
<td>0x00005003</td>
</tr>
<tr>
<td>0x010</td>
<td>Top of table</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x014 to 0xFC8</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x0FC to 0xFFC</td>
<td>ROM table registers</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>See System ROM registers</td>
</tr>
</tbody>
</table>

The second table occupies a 4-Kbyte, 32-bit wide chunk of APB-D address space, from 0xE00F0000 to 0xE00F0FFC when accessed by the debugger, and from 0x59010000 to 0x59010FFC when accessed from the system bus.

The top of each ROM table contains a number of read-only registers, including the standard CoreSight component and peripheral identity registers, see section System ROM registers.

Each debug component occupies one or more 4-Kbyte blocks of address space. This block of address space is referred to as the debug register file for the component.

The component address offset field of a ROM Table entry points to the start of the last 4-Kbyte block of the address space of the component. This block always contains the component and peripheral ID registers for the component, starting at offset 0xFD0 from the start of the block. The 4-Kbyte count field PIDR4 [7:4] specifies the number of 4-Kbyte blocks for the component.
Therefore, the process for finding the start of the address space for a component is the following:

1. Read the ROM-table entry for the component and extract its Address_Offset[18:0] from bits [31:12] of the ROM-table entry.

2. Use the address offset, together with the base address of the ROM table, ROM_Base_Address, to calculate the base address of the component:

   \[ \text{Component_Base_Address} = \text{ROM_Base_Address} + \text{Address_Offset} \]

   The Component_Base_Address is the start address of the 4-Kbyte block of the address space for the component.

3. Read the peripheral ID4 register for the component. The address of this register is:

   \[ \text{Peripheral_ID4_address} = \text{Component_Base_Address} + 0xFD0 \]

4. Extract the 4-Kbyte count field [7:4] from the value of the peripheral ID4 register.

5. Use the 4-Kbyte count field value to calculate the start address of the address space for the component. If the field value is 0b0000, that corresponds to a count value of 1, the address space for the component starts at Component_Base_Address obtained at step 2.

The topology for the CoreSight components on the APB-D is shown in Figure 759.

**Figure 759. APB-D CoreSight component topology**
For more information on the use of the ROM table, refer to the Arm® Debug Interface Architecture Specification [1].

**System ROM registers**

**SYSROM memory type register (SYSROM_MEMTYPE)**

Address offset: 0xFCC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:1  Reserved, must be kept at reset value.

Bit 0  **SYSMEM**: system memory  
0: no system memory present on this bus

**SYSROM CoreSight peripheral identity register 4 (SYSROM_PIDR4)**

Address offset: 0xFD0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **4KCOUNT[3:0]**: register file size  
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0  **JEP106CON[3:0]**: JEP106 continuation code  
0x0: STMicroelectronics JEDEC continuation code
SYSROM CoreSight peripheral identity register 0 (SYSROM_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0050

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PARTNUM[7:0]**  device part number field, bits [7:0]
- 0x50: STM32H7 device

SYSROM CoreSight peripheral identity register 1 (SYSROM_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **JEP106ID[3:0]**  JEP106 identity code field, bits [3:0]
- 0x0: STMicroelectronics JEDEC code

Bits 3:0  **PARTNUM[11:8]**  device part number field, bits [11:8]
- 0x4: STM32H7 device
SYSROM CoreSight peripheral identity register 2 (SYSROM_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 000A

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: device revision number
0x1: rev 1

Bit 3 **JEDEC**: JEDEC assigned value
1: designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x2: STMicroelectronics JEDEC code

SYSROM CoreSight peripheral identity register 3 (SYSROM_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications
SYSROM CoreSight component identity register 0 (SYSROM_CIDR0)
Address offset: 0xFF0
Reset value: 0x0000 000D

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  PREAMBLE[7:0]: component ID field, bits [7:0]
0x0D: common ID value

SYSROM CoreSight component identity register 1 (SYSROM_CIDR1)
Address offset: 0xFF4
Reset value: 0x0000 0010

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:4  CLASS[3:0]: component ID field, bits [15:12] - component class
0x1: ROM table component
Bits 3:0  PREAMBLE[11:8]: component ID field, bits [11:8]
0x0: common ID value

SYSROM CoreSight component identity register 2 (SYSROM_CIDR2)
Address offset: 0xFF8
Reset value: 0x0000 0005

Bits 31:8  Reserved, must be kept at reset value.
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[19:12]**: component ID field, bits [23:16]
0x05: common ID value

**SYSROM CoreSight component identity register 3 (SYSROM_CIDR3)**

Address offset: 0xFFC
Reset value: 0x0000 00B1

| Offset | Register name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[27:20]**: component ID field, bits [31:24]
0xB1: common ID value

**System ROM register map and reset values**

Table 521. System ROM table register map and reset values

| Offset | Register name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|

Reset values 0000 00B1
Table 521. System ROM table register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFE8</td>
<td>SYSROM_PIDR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFEC</td>
<td>SYSROM_PIDR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF0</td>
<td>SYSROM_CIDR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF4</td>
<td>SYSROM_CIDR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF8</td>
<td>SYSROM_CIDR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFFC</td>
<td>SYSROM_CIDR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

64.5.2  Cross trigger interfaces (CTI) and matrix (CTM)

The cross trigger interfaces (CTI) and cross trigger matrix (CTM) together form the CoreSight embedded cross trigger feature. There are two CTI components, one at system level and one dedicated to the Cortex-M7. The two CTIs are connected to each other via the CTM. The system-level CTI is accessible to the debugger via the system access port and associated APB-D. The Cortex-M7 CTI is physically integrated in the Cortex-M7 core and is accessible via the Cortex-M7 access port and associated AHBD.
The CTIs allow events from various sources to trigger debug and/or trace activity. For example, a transition detected on an external trigger input can start code trace or halt the processor.

Each CTI has up to 8 trigger inputs and 8 trigger outputs. Any input can be connected to any output, on the same CTI or on another CTI via the CTM.

The trigger input and output signals for each CTI are listed in Table 522 to Table 525.

Table 522. System CTI inputs

<table>
<thead>
<tr>
<th>#</th>
<th>Source signal</th>
<th>Source component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DBTRIGI</td>
<td>GPIO</td>
<td>External trigger input - allows an external signal to generate a debug event.</td>
</tr>
<tr>
<td>1</td>
<td>ETFACQCOMP</td>
<td>ETF</td>
<td>ETF capture finished - allows a debug event to be generated when the trace FIFO is empty.</td>
</tr>
<tr>
<td>2</td>
<td>ETFFULL</td>
<td>ETF</td>
<td>ETF full flag - allows a debug event to be generated when the trace FIFO is full.</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
</tbody>
</table>
### Table 522. System CTI inputs (continued)

<table>
<thead>
<tr>
<th>#</th>
<th>Source signal</th>
<th>Source component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
</tbody>
</table>

### Table 523. System CTI outputs

<table>
<thead>
<tr>
<th>#</th>
<th>Output signal</th>
<th>Destination component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DBTRIGO</td>
<td>GPIO</td>
<td>External IO trigger output - allows monitoring of events on the external DBTRIGO pin.</td>
</tr>
<tr>
<td>1</td>
<td>TPIUFLUSH</td>
<td>TPIU</td>
<td>Trace port flush trigger - causes the TPIU FIFO to be flushed.</td>
</tr>
<tr>
<td>2</td>
<td>TPIUTRIG</td>
<td>TPIU</td>
<td>Trace port enable trigger - starts trace output on the external trace port.</td>
</tr>
<tr>
<td>3</td>
<td>ETFTRIG</td>
<td>ETF</td>
<td>ETF enable trigger - starts filling the Trace FIFO.</td>
</tr>
<tr>
<td>4</td>
<td>ETFFLUSH</td>
<td>ETF</td>
<td>ETF flush trigger - causes the Trace FIFO to be flushed.</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
</tbody>
</table>

### Table 524. Cortex-M7 CTI inputs

<table>
<thead>
<tr>
<th>#</th>
<th>Source signal</th>
<th>Source component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HALTED</td>
<td>Cortex-M7 CPU</td>
<td>CPU halted - indicates CPU is in debug mode.</td>
</tr>
<tr>
<td>1</td>
<td>COMPMATCH0</td>
<td>Cortex-M7 DWT</td>
<td>DWT comparator 0 match</td>
</tr>
<tr>
<td>2</td>
<td>COMPMATCH1</td>
<td>Cortex-M7 DWT</td>
<td>DWT comparator 1 match</td>
</tr>
<tr>
<td>3</td>
<td>COMPMATCH2</td>
<td>Cortex-M7 DWT</td>
<td>DWT comparator 2 match</td>
</tr>
<tr>
<td>4</td>
<td>ETMEXTOUT0</td>
<td>Cortex-M7 ETM</td>
<td>ETM external trigger out</td>
</tr>
<tr>
<td>5</td>
<td>ETMEXTOUT1</td>
<td>Cortex-M7 ETM</td>
<td>ETM external trigger out</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
</tbody>
</table>

### Table 525. Cortex-M7 CTI outputs

<table>
<thead>
<tr>
<th>#</th>
<th>Output signal</th>
<th>Destination component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EDBGRQ</td>
<td>Cortex-M7 CPU</td>
<td>CPU halt request - puts CPU in debug mode.</td>
</tr>
<tr>
<td>1</td>
<td>nIRQ1</td>
<td>Cortex-M7 NVIC</td>
<td>Interrupt request</td>
</tr>
<tr>
<td>2</td>
<td>nIRQ2</td>
<td>Cortex-M7 NVIC</td>
<td>Interrupt request</td>
</tr>
</tbody>
</table>
There are four event channels in the cross trigger matrix, that allows up to four parallel bidirectional connections between trigger inputs and outputs on different CTIs. To connect input number \( m \) on CTI \( x \) to output number \( n \) on CTI \( y \), the input must be connected to an event channel \( p \) using the CTIINEN\( m \) register of CTI \( x \). The same channel \( p \) must be connected to the output using the CTIOUTEN\( n \) register of CTI \( y \). Note that this applies even if the input and output belong to the same CTI.

An input can be connected to more than one channel (up to four), so an input can be routed to several outputs. Similarly, an output can be connected to several inputs. It is also possible to connect several inputs/outputs to the same channel.

**Table 525. Cortex-M7 CTI outputs (continued)**

<table>
<thead>
<tr>
<th>#</th>
<th>Output signal</th>
<th>Destination component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>4</td>
<td>ETMEVENTS0</td>
<td>Cortex-M7 ETM</td>
<td>ETM trig request - enables CPU execution trace</td>
</tr>
<tr>
<td>5</td>
<td>ETMEVENTS1</td>
<td>Cortex-M7 ETM</td>
<td>ETM trig request - enables CPU execution trace</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>7</td>
<td>DBGRESTART</td>
<td>Cortex-M7 CPU</td>
<td>CPU restart request - CPU exits debug mode</td>
</tr>
</tbody>
</table>

For more information on the cross-trigger interface CoreSight component, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual [2].
CTI registers

The register file base address for each CTI is defined by the ROM table for the bus to which it is connected. The registers are the same for each CTI.

CTI control register (CTI_CONTROL)

Address offset: 0x000
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLBEN**: global enable
  0: cross-triggering disabled
  1: cross-triggering enabled

CTI trigger acknowledge register (CTI_INTACK)

Address offset: 0x010
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INTACK[7:0]**: trigger acknowledge
  There is one bit of this register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.
CTI application trigger set register (CTI_APPSET)

Address offset: 0x014
Reset value: 0x0000 0000

Bits 31:4  Reserved, must be kept at reset value.
Bits 3:0  APPSET[3:0]: set channel event
   Read:
   0bXXX0: channel 0 event inactive
   0bXXX1: channel 0 event active
   0bXX0X: channel 1 event inactive
   0bXX1X: channel 1 event active
   0bX0XX: channel 2 event inactive
   0bX1XX: channel 2 event active
   0b0XXX: channel 3 event inactive
   0b1XXX: channel 3 event active
   Write:
   0bXXX0: no effect
   0bXXX1: Sets event on channel 0.
   0bXX0X: no effect
   0bXX1X: Sets event on channel 1.
   0bX0XX: no effect
   0bX1XX: Sets event on channel 2.
   0b0XXX: no effect
   0b1XXX: Sets event on channel 3.

CTI application trigger clear register (CTI_APPCLEAR)

Address offset: 0x018
Reset value: 0x0000 0000

Bits 31:4  Reserved, must be kept at reset value.
Bits 3:0  APPCLEAR[3:0]: set channel event
   Read:
   0bXXX0: no effect
   0bXXX1: no effect
   0bXX0X: no effect
   0bXX1X: no effect
   0bX0XX: no effect
   0bX1XX: no effect
   0b0XXX: no effect
   0b1XXX: no effect
CTI application pulse register (CTI_APPPULSE)

Address offset: 0x01C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPCLEAR[3:0]**: clear channel event
- 0bXXX0: no effect
- 0bXXX1: Clears event on channel 0.
- 0bXX0X: no effect
- 0bXX1X: Clears event on channel 1.
- 0bX0XX: no effect
- 0bX1XX: Clears event on channel 2.
- 0b0XXX: no effect
- 0b1XXX: Clears event on channel 3.

**CTI trigger IN x enable register (CTI_INENx)**

Address offset: 0x020 + 4 * x, where x = 0 to 7
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPPULSE[3:0]**: pulse channel event

- This register clears itself immediately.
- 0bXXX0: no effect
- 0bXXX1: Generates pulse on channel 0.
- 0bXX0X: no effect
- 0bXX1X: Generates pulse on channel 1.
- 0bX0XX: no effect
- 0bX1XX: Generates pulse on channel 2.
- 0b0XXX: no effect
- 0b1XXX: Generates pulse on channel 3.
CTI trigger OUT x enable register (CTI_OUTENx)

Address offset: 0xA0 + 4 * x, where x = 0 to 7
Reset value: 0x0000 0000

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  TRIGINEN[3:0]: cross-trigger event enable
Enables or disables a cross-trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).
  0bXXX0: Trigger n does not generate events on channel 0.
  0bXXX1: Trigger n generates events on channel 0.
  0bX0X0: Trigger n does not generate events on channel 1.
  0bX0X1: Trigger n generates events on channel 1.
  0bX1XX: Trigger n does not generate events on channel 2.
  0bX1XX: Trigger n generates events on channel 2.
  0b0XXX: Trigger n does not generate events on channel 3.
  0b1XXX: Trigger n generates events on channel 3.

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  TRIGOUTEN[3:0]: enable trigger upon event
For each channel, the field defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).
  0bXXX0: Channel 0 events do not generate triggers on trigger output n.
  0bXXX1: Channel 0 events generate triggers on trigger output n.
  0bXX0X: Channel 1 events do not generate triggers on trigger output n.
  0bXX1X: Channel 1 events generate triggers on trigger output n.
  0bX0XX: Channel 2 events do not generate triggers on trigger output n.
  0bX1XX: Channel 2 events generate triggers on trigger output n.
  0b0XXX: Channel 3 events do not generate triggers on trigger output n.
  0b1XXX: Channel 3 events generate triggers on trigger output n.
## CTI trigger IN status register (CTI_TRGISTS)

Address offset: 0x130  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **TRIGINSTATUS[7:0]:** trigger input status  
There is one bit of the register for each CTITRIGIN input. When a bit is set to 1, it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

## CTI trigger OUT status register (CTI_TRGOSTS)

Address offset: 0x134  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **TRIGOUTSTATUS[7:0]:** trigger output status  
There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1, it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.
**CTI channel IN status register (CTI_CHINSTS)**

Address offset: 0x138  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

### Bits 31:4
Reserved, must be kept at reset value.

### Bits 3:0 **CHINSTATUS[3:0]**: channel input status

There is one bit of the register for each channel input. When a bit is set to 1, it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

---

**CTI channel OUT status register (CTI_CHOUTSTS)**

Address offset: 0x13C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

### Bits 31:4
Reserved, must be kept at reset value.

### Bits 3:0 **CHOUTSTATUS[3:0]**: channel output status

There is one bit of the register for each channel output. When a bit is set to 1, it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.
CTI channel gate register (CTI_GATE)

Address offset: 0x140
Reset value: 0x0000 000F

Bits 31:4  Reserved, must be kept at reset value.
Bits 3:0  GATEEN[3:0]: channel output enable
  For each channel, this field defines whether an event on that channel can propagate over the
  CTM to other CTIs.
  0bXXX0: Channel 0 events do not propagate.
  0bXXX1: Channel 0 events propagate.
  0bXX0X: Channel 1 events do not propagate.
  0bXX1X: Channel 1 events propagate.
  0bX0XX: Channel 2 events do not propagate.
  0bX1XX: Channel 2 events propagate.
  0b0XXX: Channel 3 events do not propagate.
  0b1XXX: Channel 3 events propagate.

CTI claim tag set register (CTI_CLAIMSET)

Address offset: 0xFA0
Reset value: 0x0000 000F

Bits 31:4  Reserved, must be kept at reset value.
Bits 3:0  CLAIMSET[3:0]: Set claim tag bits
  Write:
  0000: no effect
  0xx1: Sets bit 0.
  1x1x: Sets bit 1.
  1xxx: Sets bit 3.
  Read:
  0xF: Indicates there are four bits in claim tag.
CTI claim tag clear register (CTI_CLAIMCLR)

Address offset: 0xFA4
Reset value: 0x0000 0000

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  `CLAIMCLR[3:0]`: reset claim tag bits
    Write:
    0000: no effect
    xxx1: Clears bit 0.
    xx1x: Clears bit 1.
    x1xx: Clears bit 2.
    1xxx: Clears bit 3.
    Read: Returns current value of claim tag.

CTI lock access register (CTI_LAR)

Address offset: 0xFB0
Reset value: 0xXXXX XXXX

Bits 31:0  `ACCESS_W[31:0]`: CTI register write access enable
    This field enables write access to some CTI registers by processor cores (debuggers do not need to unlock the component).
    0xC5ACCE55: write access enabled
    Other values: write access disabled
CTI lock status register (CTI_LSR)
Address offset: 0xFB4
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:3   Reserved, must be kept at reset value.

Bit 2  **LOCKTYPE**: size of the CTI_LAR register
   0: 32-bit

Bit 1  **LOCKGRANT**: current status of lock
   This bit always returns zero when read by an external debugger.
   0: write access permitted
   1: write access blocked. Only read access is permitted.

Bit 0  **LOCKEXIST**: existence of lock control mechanism
   The bit indicates whether a lock control mechanism exists. It always returns zero when read
   by an external debugger.
   0: no lock control mechanism
   1: lock control mechanism implemented

CTI authentication status register (CTI_AUTHSTAT)
Address offset: 0xFB8
Reset value: 0x0000 000A

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8   Reserved, must be kept at reset value.

Bits 7:6   **SNID[1:0]**: security level for secure non-invasive debug
   0x0: not implemented
Bits 5:4  **SID[1:0]**: security level for secure invasive debug
0x0: not implemented

Bits 3:2  **NSID[1:0]**: security level for non-secure non-invasive debug
0x2: disabled
0x3: enabled

Bits 1:0  **NSID[1:0]**: security level for non-secure invasive debug
0x2: disabled
0x3: enabled

**CTI device configuration register (CTI_DEVID)**

Address offset: 0xFC8

Reset value: 0x0004 0800

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:20  Reserved, must be kept at reset value.

Bits 19:16  **NUMCH[3:0]**: number of ECT channels available
0x4: four channels

Bits 15:8  **NUMTRIG[7:0]**: number of ECT triggers available
0x8: height trigger inputs and height trigger outputs

Bits 7:5  Reserved, must be kept at reset value.

Bits 4:0  **EXTMUXNUM[4:0]**: number of trigger input/output multiplexers
0x0: none

**CTI device type identifier register (CTI_DEVTYPE)**

Address offset: 0xFCC

Reset value: 0x0000 0014

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**SUBTYPE[3:0]**

**MAJORTYPE[3:0]**
Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: sub-classification
0x1: Indicates that this component is a cross-triggering component.

Bits 3:0 **MAJORTYPE[3:0]**: major classification
0x4: Indicates that this component allows a debugger to control other components in a CoreSight SoC-400 system.

**CTI CoreSight peripheral identity register 4 (CTI_PIDR4)**

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

**CTI CoreSight peripheral identity register 0 (CTI_PIDR0)**

Address offset: 0xFE0
Reset value: 0x0000 0006

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number field, bits [7:0]
0x06: CTI part number
**CTI CoreSight peripheral identity register 1 (CTI_PIDR1)**

Address offset: 0xFE4  
Reset value: 0x0000 00B9

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]  
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number field, bits [11:8]  
0x9: CTI part number

**CTI CoreSight peripheral identity register 2 (CTI_PIDR2)**

Address offset: 0xFE8  
Reset value: 0x0000 004B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number  
0x4: r0p5

Bit 3 **JEDEC**: JEDEC assigned value  
1: designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]  
0x3: Arm® JEDEC code
### CTI CoreSight peripheral identity register 3 (CTI_PIDR3)

Address offset: 0xFEC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>REVAND[3:0]</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version  
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified  
0x0: no customer modification

### CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0  
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>Bit 31:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>PREAMBLE[7:0]</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID field, bits [7:0]  
0x0D: common ID value

### CTI CoreSight component identity register 1 (CTI_CIDR1)

Address offset: 0xFF4  
Reset value: 0x0000 0090

<table>
<thead>
<tr>
<th>Bit 31:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>CLASS[3:0]</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CLASS[3:0]**: PREAMBLE[11:8]  
0x0: no metal fix
**CTI CoreSight component identity register 2 (CTI_CIDR2)**

Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>Address</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:4  **CLASS[3:0]**: component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0  **PREAMBLE[11:8]**: component ID field, bits [11:8]
0x0: common ID value

**CTI CoreSight component identity register 3 (CTI_CIDR3)**

Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>Address</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  **PREAMBLE[19:12]**: component ID field, bits [23:16]
0x05: common ID value

**CTI CoreSight component identity register 4 (CTI_CIDR4)**

Address offset: 0xFC0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  **PREAMBLE[27:20]**: component ID field, bits [31:24]
0xB1: common ID value
## CTI register map and reset values

### Table 526. CTI register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>CTI_CONTROL</td>
<td>0x000</td>
<td>CTI_INTACK</td>
<td>0x000</td>
<td>CTI_APPSET</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>CTI_INTACK</td>
<td>0x000</td>
<td>CTI_APPSET</td>
<td>0x000</td>
<td>CTI_APPCLEAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x018</td>
<td>CTI_APPCLEAR</td>
<td>0x000</td>
<td>CTI_APPSET</td>
<td>0x000</td>
<td>CTI_APPPULSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01C</td>
<td>CTI_APPPULSE</td>
<td>0x000</td>
<td>CTI_APPSET</td>
<td>0x000</td>
<td>CTI_APPCLEAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x020</td>
<td>CTI_INEN0</td>
<td>0x000</td>
<td>CTI_APPSET</td>
<td>0x000</td>
<td>CTI_APPPULSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>CTI_INEN7</td>
<td>0x000</td>
<td>CTI_APPSET</td>
<td>0x000</td>
<td>CTI_APPPULSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x03C</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x040</td>
<td>Reserved</td>
<td>0x000</td>
<td>CTI_APPSET</td>
<td>0x000</td>
<td>CTI_APPPULSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>Reserved</td>
<td>0x000</td>
<td>CTI_APPSET</td>
<td>0x000</td>
<td>CTI_APPPULSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x05C</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x060</td>
<td>Reserved</td>
<td>0x000</td>
<td>CTI_APPSET</td>
<td>0x000</td>
<td>CTI_APPPULSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>Reserved</td>
<td>0x000</td>
<td>CTI_APPSET</td>
<td>0x000</td>
<td>CTI_APPPULSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0BC</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x120</td>
<td>Reserved</td>
<td>0x000</td>
<td>TRIGINSTS</td>
<td>0x000</td>
<td>TRIGOUTSTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x12C</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x130</td>
<td>CTI_TRIGISTS</td>
<td>0x000</td>
<td>TRIGINSTS</td>
<td>0x000</td>
<td>TRIGOUTSTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x134</td>
<td>CTI_TRIGOSTS</td>
<td>0x000</td>
<td>TRIGINSTS</td>
<td>0x000</td>
<td>TRIGOUTSTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x138</td>
<td>CTI_CHINSTS</td>
<td>0x000</td>
<td>TRIGINSTS</td>
<td>0x000</td>
<td>TRIGOUTSTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 526. CTI register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x13C</td>
<td>CTI_CHOUTSTS</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
<td>CTI_GATE</td>
<td>1 1 1 1 1 1</td>
</tr>
<tr>
<td>0x140</td>
<td>CTI_GATE</td>
<td>0x140</td>
<td>CTI_CLAIMSET</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0x144 to 0xF9C</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x144 to 0xFAC</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFA0</td>
<td>CTI_CLAIMCLR</td>
<td>0x140</td>
<td>CTI_CLAIMCLR</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0xFA4</td>
<td>CTI_CLAIMCLR</td>
<td>0x140</td>
<td>CTI_CLAIMCLR</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0xFA8 to 0xFAC</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0xFA8 to 0xFAC</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFB0</td>
<td>CTI_LAR</td>
<td>0x140</td>
<td>CTI_LAR</td>
<td>0x140</td>
<td></td>
</tr>
<tr>
<td>0xFB4</td>
<td>CTI_LSR</td>
<td>0x140</td>
<td>CTI_LSR</td>
<td>0x140</td>
<td></td>
</tr>
<tr>
<td>0xFB8</td>
<td>CTI_AUTHSTAT</td>
<td>0x140</td>
<td>CTI_AUTHSTAT</td>
<td>0x140</td>
<td></td>
</tr>
<tr>
<td>0xFC8</td>
<td>CTI_DEVID</td>
<td>0x140</td>
<td>CTI_DEVID</td>
<td>0x140</td>
<td></td>
</tr>
<tr>
<td>0xFCC</td>
<td>CTI_DEVTYPE</td>
<td>0x140</td>
<td>CTI_DEVTYPE</td>
<td>0x140</td>
<td></td>
</tr>
</tbody>
</table>

Reset values are shown for each register, with offsets for each bit field provided.
64.5.3 Trace funnel (CSTF)

The trace funnel is a CoreSight component that combines the ATB buses from two trace sources into one single ATB. The CSTF has two ATB slave ports, and one ATB master port. An arbiter selects the slave ports according to a programmable priority.

The slave ports are connected as follows:

- S0: Cortex-M7 ETM
- S1: Cortex-M7 ITM
The CSTF registers allow the slave ports to be individually enabled, and their priority settings to be configured. The priorities can be modified only when trace is disabled. The arbitration works as follows:

- The arbiter selects the slave port with the highest assigned priority that has data valid
- Up to $min\_hold\_time$ transfers are passed from the selected slave to the master port, where $min\_hold\_time$ is programmable in the CONTROL register.
- A new arbitration is then performed

High priority should be assigned to slave ports connected to sources with a small amount of buffering, or where data loss can not be tolerated. Low priority should be assigned to less critical sources or those with large buffers.

For more information on the ATB Funnel CoreSight component, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual [2].

**Trace funnel registers**

**CSTF control register (CSTF_CTRL)**

Address offset: 0x000

Reset value: 0x0000 0300

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 $MIN\_HOLD\_TIME[3:0]$: number of transactions between arbitrations

- 0x0: 1 transaction
- 0xE: 15 transactions
- 0xF: reserved

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **ENS1**: slave port S1 enable

- 0: S1 port disabled
- 1: S1 port enabled

Bit 0 **ENS0**: slave port S0 enable

- 0: S0 port disabled
- 1: S0 port enabled
CSTF priority register (CSTF_PRIORITY)

Address offset: 0x004
Reset value: 0x0000 0688

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:3 PRIPORT1[2:0]: slave port S1 priority
0: highest priority
7: lowest priority

Bits 2:0 PRIPORT0[2:0]: Slave port S0 priority
0: highest priority
7: lowest priority

CSTF claim tag set register (CSTF_CLAIMSET)

Address offset: 0xFA0
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 CLAIMSET[3:0]: set claim tag bits
Write:
0000: no effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.

Read:
0xF: Indicates there are four bits in claim tag;
CSTF claim tag clear register (CSTF_CLAIMCLR)

Address offset: 0xFA4
Reset value: 0x0000 0000

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  CLAIMCLR[3:0]: reset claim tag bits
  Write:
  0000: no effect
  xxx1: Clears bit 0.
  xx1x: Clears bit 1.
  x1xx: Clears bit 2.
  1xxx: Clears bit 3.
  Read: Returns current value of claim tag.

CSTF lock access register (CSTF_LAR)

Address offset: 0xFB0
Reset value: 0xXXXX XXXX

Bits 31:0  ACCESS_W[31:0]: CSTF register write access enable
  The field enables write access to some CSTF registers by processor cores (debuggers do not need to unlock the component).
  0xC5ACCE55: write access enabled
  Other values: write access disabled
### CSTF lock status register (CSTF_LSR)

Address offset: 0xFB4  
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:3 Reserved, must be kept at reset value.

- **Bit 2 LOCKTYPE**: size of the CSTF_LAR register  
  0: 32-bit

- **Bit 1 LOCKGRANT**: current status of lock  
  This bit always returns zero when read by an external debugger.  
  0: write access permitted  
  1: write access blocked. Only read access is permitted.

- **Bit 0 LOCKEXIST**: existence of lock control mechanism  
  This bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.  
  0: no lock control mechanism  
  1: lock control mechanism implemented

### CSTF authentication status register (CSTF_AUTHSTAT)

Address offset: 0xFB8  
Reset value: 0x0000 000A

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Bits 31:8** Reserved, must be kept at reset value.

- **Bits 7:6** **SNID[1:0]**: security level for secure non-invasive debug  
  0x0: not implemented
Bits 5:4  **SID[1:0]**: security level for secure invasive debug  
0x0: not implemented  
Bits 3:2  **NSNID[1:0]**: security level for non-secure non-invasive debug  
0x2: disabled  
0x3: enabled  
Bits 1:0  **NSID[1:0]**: security level for non-secure invasive debug  
0x2: disabled  
0x3: enabled

**CSTF CoreSight device identity register (CSTF_DEVID)**

Address offset: 0xFC8  
Reset value: 0x0000 0024

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.  
Bits 7:4  **SCHEME[3:0]**: priority scheme  
0x2: static priority  
Bits 3:0  **PORTCNT[3:0]**: number of input ports connected  
0x4: four input ports

**CSTF CoreSight device type identity register (CSTF_DEVTYPE)**

Address offset: 0xFCC  
Reset value: 0x0000 0012

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.  
Bits 7:0  **DEVTYPEID[7:0]**: device type identifier  
0x12: trace funnel
CSTF CoreSight peripheral identity register 4 (CSTF_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **4KCOUNT[3:0]**: register file size

0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0  **JEP106CON[3:0]**: JEP106 continuation code

0x4: Arm® JEDEC code

CSTF CoreSight peripheral identity register 0 (CSTF_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0008

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PARTNUM[7:0]**: part number field, bits [7:0]

0x08: CSTF part number
CSTF CoreSight peripheral identity register 1 (CSTF_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 00B9

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:4</td>
<td>JEP106ID[3:0]: JEP106 identity code field, bits [3:0]</td>
</tr>
<tr>
<td></td>
<td>0xB: Arm® JEDEC code</td>
</tr>
<tr>
<td>Bits 3:0</td>
<td>PARTNUM[11:8]: part number field, bits [11:8]</td>
</tr>
<tr>
<td></td>
<td>0x9: CSTF part number</td>
</tr>
</tbody>
</table>

CSTF CoreSight peripheral identity register 2 (CSTF_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 001B

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:4</td>
<td>REVISION[3:0]: component revision number</td>
</tr>
<tr>
<td></td>
<td>0x1: r0p1</td>
</tr>
<tr>
<td>Bit 3</td>
<td>JEDEC: JEDEC assigned value</td>
</tr>
<tr>
<td></td>
<td>1: designer ID specified by JEDEC</td>
</tr>
<tr>
<td></td>
<td>0x3: Arm® JEDEC code</td>
</tr>
</tbody>
</table>
**CSTF CoreSight peripheral identity register 3 (CSTF_PIDR3)**

Address offset: 0xFEC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version  
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified  
0x0: no customer modification

**CSTF CoreSight component identity register 0 (CSTF_CIDR0)**

Address offset: 0xFF0  
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID field, bits [7:0]  
0x0D: common ID value

**CSTF CoreSight component identity register 1 (CSTF_CIDR1)**

Address offset: 0xFF4  
Reset value: 0x0000 0090

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 24:16 **CLASS[3:0]** PREAMBLE[11:8]  
0x0D: common ID value
CSTF CoreSight component identity register 2 (CSTF_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

CSTF CoreSight component identity register 3 (CSTF_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1
Trace funnel register map and reset values

### Table 527. CSTF register map and reset values

| Offset | Register name          | Reset value | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|------------------------|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x000  | CSTF_CTRL              |             | 0   | 0   | 1   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x004  | CSTF_PRIORITY          |             |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x008  | Reserved               | 0           | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0xFA0  | CSTF_CLAIMSET          |             |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xFA4  | CSTF_CLAIMCLR          |             |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xFB0  | Reserved               | 0           | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   |
| 0xFB4  | CSTF_LAR               | ACCESS_W[31:0] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xFB8  | CSTF_LSR               |             |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xFC0  | Reserved               | 0           | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0xFC8  | CSTF_DEVID             |             |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xFCC  | CSTF_DEVTYPE           |             |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
64.5.4 Embedded trace FIFO (ETF)

The ETF is an 8-Kbyte memory that captures trace data from two trace sources, ETM and ITM, of the CPU core. The ETF is a design configuration of the CoreSight trace memory controller component.

The ETF can be used in three modes (selected in the mode register):
1. Hardware FIFO mode

   The trace memory is used as a FIFO that is drained through the ATB master interface. Trace data is captured into the trace RAM and when full, the incoming trace stream is...
stalled. When the trace buffer is not empty, trace data is drained out through the ATB master interface to the TPIU.

In this mode, the role of the FIFO is to smooth the flow of trace information arriving at the trace port. Since the trace data can be very bursty in nature, the peak data rate can easily exceed the port capability, resulting in an overflow. The ETF allows a steady data rate at the trace port, that can then be sized according to the average rate rather than the peak. The trace is stored off-chip in real time by the trace port analyzer tool, and so the trace log can be very big.

2. Software FIFO mode

The trace memory is used as a FIFO that can be read through the ETF_RRD register while trace is being captured. Trace data is captured into the trace RAM and when full, the incoming trace stream is stalled.

This mode allows the trace to be transferred by DMA into the system memory, or to a high-speed interface (such as SPI or USB), or even monitored by software running on one of the cores. Note that unlike the hardware FIFO mode, this mode is invasive, since it uses system resources that are shared by the processor.

3. Circular buffer mode

The trace memory is used as a circular buffer. Trace data is captured into the trace memory starting from the location pointed to by the write pointer register. Even when the trace memory is full, incoming trace data continues to be overwritten into the trace memory until a stop condition occurs.

In this mode, the ETF stores the trace data on-chip, so the trace log size is limited to that of the ETF SRAM, 8 Kbytes in this case. Being a circular buffer, if the FIFO becomes full, incoming trace data overwrites the oldest stored data and the oldest stored data is lost. Therefore the content of the trace buffer represents the most recent activity of the processor, up to the point when the buffer was stopped, rather than all the activity since the trace was started.

There are three possible methods to read out the buffer contents once the trace stops:

– via the trace port - with the TPIU enabled, the content of the buffer is output over the trace port. This can be done by setting the DRAINBUF bit in the ETF_FFCR register.
– via the debug port - the debugger can read the buffer via the RRD register that is accessible over the system APB-D.
– by software - the processor can read the buffer via the RRD register, since the APB-D is accessible from the system bus.
The ETF transition between the following states are described below:

- **Disabled**
  This state is entered after a reset or when trace capture is disabled. The ETF must only be programmed in this state.

- **Running**
  Trace capture is performed in this state. It is entered by enabling trace capture while in Disabled state.

- **Stopped**
  Trace capture is stopped in this state but the content of the buffer can be read out or drained. This state is entered after a stop event (trigger or flush).

- **Disabling**
  This is a transitional state while disabling trace capture.

- **Stopping**
  This is a transitional state while stopping trace capture.

- **Draining**
  This state is entered while draining the buffer in Stopped state.

The state transition diagram is shown in Figure 762.

For more information on the trace memory controller CoreSight component, refer to the Arm® CoreSight trace memory controller technical reference manual [3].
ETF registers

ETF RAM size register (ETF_RSZ)
Address offset: 0x004
Reset value: 0x0000 0800

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RSZ[30:16]

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

RSZ[15:0]

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  Reserved, must be kept at reset value.

Bits 30:0  RSZ[30:0]: RAM size
The value of the field indicates the number of 32-bit words.
0x800: 2048 words = 8 Kbyte

ETF status register (ETF_STS)
Address offset: 0x00C
Reset value: 0x0000 001C

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

EMPTY .Formatter empty

This bit is set when trace capture has stopped and all internal pipelines and buffers have drained. Unlike the READY bit, it is not affected by buffer drains. The ACQCMP output reflects the value of this bit.
Bit 2 **READY**: ETF ready
This bit is set when trace capture has stopped and all internal pipelines and buffers have drained (Stopped or Disabled state).

Bit 1 **TRIGD**: triggered
This bit is set when trace capture is in progress and the TMC has detected a trigger event. This bit is cleared when leaving Disabled state. This bit is operational only in the circular buffer mode. In all other modes, this bit is always low. This bit does not indicate that a trigger has been embedded in the formatted output trace data from the TMC. Trigger indication on the output trace stream is determined by the programming of the formatter and flush control register, ETF_FFCR.

Bit 0 **FULL**: Trace buffer full
In circular buffer mode, this flag is set when the RAM write pointer wraps around the top of the buffer, and remains set until the TCEN bit of the ETF_CTL register is cleared and set. In software and hardware FIFO modes, this flag indicates that the current space in the trace memory is less than or equal to the value programmed in the ETF_BUFWM register, that is, fill level >= MEM_SIZE - BUFWM. This bit is cleared when leaving Disabled state. The FULL output reflects the value of this register bit.

**ETF RAM read data register (ETF_RRD)**

Address offset: 0x010
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

RRD[31:16]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

RRD[15:0]

Bits 31:0 **RRD[31:0]**: RAM read data.
Circular buffer mode:
When in Stopped state and the buffer is not empty, reading this register returns the next word of data from the trace buffer. When all of the trace buffer has been read, the Empty bit in the ETF_STS Register is set, and subsequent reads return 0xFFFFFFF. Reading this register when not in Stopped state returns 0xFFFFFFFF.

Software FIFO mode:
Reading this register returns data from the FIFO. If this register is read when the FIFO is empty, the data returned is 0xFFFFFFFF.

Hardware FIFO mode:
Reading this register returns 0xFFFFFFFF.
ETF RAM read pointer register (ETF_RRP)

Address offset: 0x014
Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:13  Reserved, must be kept at reset value.

Bits 12:0  **RRP[12:0]:** RAM read pointer

This register contains the value of the read pointer that is used to read entries from the trace memory over the APB interface via the ETF_RRD register. The pointer can be programmed with a byte address, 64-bit aligned (that is, bits 0 to 3 must be zero). The pointer is incremented by 8 each time a full 64-bit FIFO entry has been written. When the pointer reaches its maximum value, it wraps around.

This register can only be written in Disabled state. It can be read in Disabled state, in Stopped state in circular buffer mode and SW FIFO mode, and also in Running and Stopping states in SW FIFO mode.

ETF RAM write pointer register (ETF_RWP)

Address offset: 0x018
Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:13  Reserved, must be kept at reset value.

Bits 12:0  **RWP[12:0]:** RAM write pointer

This register contains the value of the write pointer that is used to write entries into the trace memory over the APB interface via the ETF_RWD register. The pointer can be programmed with a byte address, 64-bit aligned (that is, bits 0 to 3 must be zero). The pointer is incremented by 8 each time a full 64-bit FIFO entry has been read. When the pointer reaches its maximum value, it wraps around.

This register can only be written in Disabled state. It can be read in Disabled state, in Stopped state in circular buffer mode and SW FIFO mode, and also in Running and Stopping states in SW FIFO mode.
ETF trigger counter register (ETF_TRG)
Address offset: 0x01C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:11  Reserved, must be kept at reset value.
Bits 10:0  TRG[10:0]: trigger counter
In circular buffer mode, this field specifies the number of 32-bit words to capture in the trace RAM following the detection of either a rising edge on the TRIGIN input or a trigger packet in the incoming trace stream, ATID =7'h7D. On capturing the specified number of data words, a trigger event occurs. The effect of a trigger event on the ETF behavior is controlled by the FFCR Register.

The number of 32-bit words written into the trace RAM following the trigger is the value stored in this register, plus one. This register is ignored when the ETF is in software FIFO mode or hardware FIFO mode. When the trigger counter starts counting, any additional triggers, either on TRIGIN or in the incoming trace stream, are ignored until the counter reaches zero. When the trigger counter has reached zero, it remains at zero until it is re-programmed with a write to this register.

This register is cleared when the READY bit of ETF_STS register goes high, so that the state of the counter when trace capture has stopped, does not affect a subsequent trace capture session. Writing to this register when not in Disabled state, results in unpredictable behavior. A read access to this register is permitted at any time when in Disabled state or in circular buffer mode. A read access returns the current value of the trigger counter.

ETF control register (ETF_CTL)
Address offset: 0x020
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TCEN
Bits 31:1 Reserved, must be kept at reset value.

Bit 0  **TCEN**: trace capture enable
When writing:
0: trace capture disabled (moves from Running, Stopping or Stopped state into Disabling or Disabled state)
1: trace capture enabled (moves from Disabled state into Running state)
When reading, this bit is low when in Disabling or Disabled states, and high otherwise.

**ETF RAM write data register (ETF_RWD)**
Address offset: 0x024
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **RWD[31:0]**: RAM write data
When in Disabled state, a write to this register stores data at the location pointed to by the RWP. Writes to this register when not in Disabled state are ignored. When a full memory width (64-bit) of data has been written, the data is written to memory and the RAM write pointer is incremented to the next memory word.
This register is used for test purposes.

**ETF mode register (ETF_MODE)**
Address offset: 0x028
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **MODE[1:0]**
When in Disabled state, writes to this register are ignored. Reads from this register are low.
This register is used for test purposes.
ETF latched buffer fill level register (ETF_LBUFLVL)

Address offset: 0x02C

Reset value: 0x0000 0000

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LBUFLEVEL[11:0]**: latched buffer fill level

Reading this register returns the maximum fill level of the trace memory in 32-bit words since this register was last read. Reading this register also results in its contents being updated to the current fill level.

When entering Disabled state, this register retains its last value. While in Disabled state, reads from this register do not affect its value. When exiting Disabled state, the LB bufLEVEL register is cleared.

This register is used for performance analysis of the trace system.
ETF current buffer fill level register (ETF_CBUFLVL)

Address offset: 0x030
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES.

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CBUFLEVEL[11:0]</td>
<td></td>
</tr>
<tr>
<td>r r r r r r r r r r r r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:12  Reserved, must be kept at reset value.
Bits 11:0  CBUFLEVEL[11:0]: current buffer fill level
Reading this register returns the current fill level of the trace memory in 32-bit words.
This register is cleared when TCEN is low.

ETF buffer level watermark register (ETF_BUFWM)

Address offset: 0x034
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES.

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES. RES.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFWM[10:0]</td>
<td></td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:11  Reserved, must be kept at reset value.
Bits 10:0  BUFWM[10:0]: buffer level watermark
The value programmed into this register indicates the required threshold vacancy level (in 32-bit words) in the trace memory. When the space in the FIFO is less than or equal to this value (that is Fill level >= MEM_SIZE - BUFWM), the FULL output is pulled high and the FULL bit in the STS register is set.
This register is used only in software FIFO and hardware FIFO modes. In circular buffer mode, this functionality can be obtained by programming the RWP to the required vacancy trigger level, so that, when the pointer wraps around, the FULL bit is set indicating that the vacancy level has fallen below the required level.
The maximum value that can be written into this register is MEM_SIZE - 1. In this case, the FULL bit output is asserted after the first 32-bit word is written to trace memory.
Writing to this register other than when in disabled state results in unpredictable behavior.
### ETF formatter and flush status register (ETF_FFSR)

Address offset: 0x300  
Reset value: 0x0000 0002

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:2  Reserved, must be kept at reset value.

- **Bit 1 FTSTOPPED**: formatter stopped  
  This bit behaves in the same way as the FEMPTY bit in the ETF_STS register.

- **Bit 0 FLINPROG**: flush in progress  
  this bit indicates whether a flush on the ATB slave port is in progress. It reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the ETF_FFCR register, or requested by the ATB master port.  
  0: no flush in progress  
  1: flush in progress

### ETF formatter and flush control register (ETF_FFCR)

Address offset: 0x304  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 16 | 15 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| DRAINBUF | STPONTRGEV | STOPONFL | TRIGONFL | TRGONTRGEV | TRGONTRGIN | Res | FLUSHMAN | FONTRGEV | FONFLIN | Res | Res | Res | Res | ENTI | ENFT |

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

*ST*
Bits 31:15  Reserved, must be kept at reset value.

Bit 14  **DRAINBUF**: drain buffer

This bit is used to enable draining of the trace data through the ATB master interface after the formatter has stopped. This is useful in circular buffer mode to capture trace data into trace memory and then to drain the captured trace through the ATB master interface.

Writing 1 to this bit when in Stopped state starts the drain of the contents of the trace buffer through the ATB master interface. This bit always reads as zero. The READY bit in the ETF_STS register goes low while the drain is in progress.

This bit is functional only when the ETF is in circular buffer mode and formatting is enabled, that is, the ENFT bit in the ETF_FFCR register is set. Setting this bit when the ETF is in any other mode or when not in Stopped state, results in unpredictable behavior.

When trace capture is complete in circular buffer mode, all the captured trace must be retrieved from the trace memory through the same mechanism, either read all trace data out through RRD reads, or drain all trace data by setting the DRAINBUF bit. Setting the DRAINBUF bit after some of the captured trace has been read out through RRD, results in unpredictable behavior.

Bit 13  **STPONTRGEV**: stop on trigger event

0: no effect
1: Stops trace capture when a trigger event occurs.

Enabling the ETF in software FIFO mode or hardware FIFO mode with this bit set, results in unpredictable behavior.

Bit 12  **STOPONFL**: stop on flush

0: no effect
1: Stops trace capture when flush is completed.

If a flush is initiated by the ATB master interface, its completion does not lead to a formatter stop regardless of the value programmed in this bit.

Bit 11  Reserved, must be kept at reset value.

Bit 10  **TRIGONFL**: trigger on flush

0: no effect
1: Indicates a trigger in the trace stream when flush is completed.

If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.

If a flush is initiated by the ATB master interface, its completion does not lead to a trigger indication regardless of the value programmed in this bit.

Bit 9  **TRGONTRGEV**: trigger on trigger event

0: No effect
1: Indicates a trigger in the trace stream when trigger event occurs.

If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.

This bit is not supported in software FIFO mode or hardware FIFO mode.

Bit 8  **TRGONTRGIN**: trigger on trigger in

0: no effect
1: Indicates a trigger in the trace stream when a rising edge is detected on the TRIGIN input.

If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.

Bit 7  Reserved, must be kept at reset value.
ETF periodic synchronization counter register (ETF_PSCR)

Address offset: 0x308
Reset value: 0x0000 000A

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>FLUSHMAN: manual flush</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no effect</td>
</tr>
<tr>
<td>1</td>
<td>Flushes the trace FIFO and pipeline. This bit is cleared automatically when the flush completes. If the TCEN bit in the ETF_CTL register is 0, writes to this bit are ignored.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 5</th>
<th>FONTRGEV: flush on trigger event</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no effect</td>
</tr>
<tr>
<td>1</td>
<td>Flushes the trace FIFO and pipeline if a trigger event occurs. This bit is not supported in software FIFO mode or hardware FIFO mode. If STPONTRGEV is set, this bit is ignored.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>FONFLIN: flush on flush in</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no effect</td>
</tr>
<tr>
<td>1</td>
<td>Flushes the trace FIFO and pipeline when a rising edge is detected on the FLUSHIN input.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 3:2</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>ENTI: Enable trigger insertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting this bit enables the insertion of triggers in the formatted trace stream. A trigger is indicated by inserting one byte of data 8'h00 with ATID 7'h7D in the trace stream. Trigger indication on the trace stream is additionally controlled by the register bits TRIGONFL, TRGONTRGEV, and TRGONTRGIN in the ETF_FFCR register. This bit can only be changed when READY is high, and TCEN is low. This bit takes effect only when the ENFT bit in this register is set. If the ENTI bit is set to high when ENFT is low, it results in formatting being enabled.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>ENFT: enable formatting</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>formatting disabled. Incoming trace data is assumed to be from a single trace source.</td>
</tr>
<tr>
<td>1</td>
<td>formatting enabled. If multiple ATIDs are received by the ETF when trace capture is enabled and the formatter is disabled, it results in interleaving of trace data. Disabling of formatting is supported only in circular buffer mode. If the ETF is enabled in a mode other than circular buffer mode with ENFT low, it results in formatting being enabled. If the ENTI bit is set to high when ENFT is low, it results in formatting being enabled. This bit is ignored when in Disabled state.</td>
</tr>
</tbody>
</table>

### Table

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res.</td>
</tr>
<tr>
<td>30</td>
<td>Res.</td>
</tr>
<tr>
<td>29</td>
<td>Res.</td>
</tr>
<tr>
<td>28</td>
<td>Res.</td>
</tr>
<tr>
<td>27</td>
<td>Res.</td>
</tr>
<tr>
<td>26</td>
<td>Res.</td>
</tr>
<tr>
<td>25</td>
<td>Res.</td>
</tr>
<tr>
<td>24</td>
<td>Res.</td>
</tr>
<tr>
<td>23</td>
<td>Res.</td>
</tr>
<tr>
<td>22</td>
<td>Res.</td>
</tr>
<tr>
<td>21</td>
<td>Res.</td>
</tr>
<tr>
<td>20</td>
<td>Res.</td>
</tr>
<tr>
<td>19</td>
<td>Res.</td>
</tr>
<tr>
<td>18</td>
<td>Res.</td>
</tr>
<tr>
<td>17</td>
<td>Res.</td>
</tr>
<tr>
<td>16</td>
<td>Res.</td>
</tr>
<tr>
<td>15</td>
<td>PSCOUNT[4:0]</td>
</tr>
<tr>
<td>14</td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>rw</td>
</tr>
</tbody>
</table>

2808/2974 RM0455 Rev 3
ETF claim tag set register (ETF_CLAIMSET)

Address offset: 0xFA0
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 CLAIMSET[3:0]: set claim tag bits
Write:
0000: no effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

ETF claim tag clear register (ETF_CLAIMCLR)

Address offset: 0xFA4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 CLAIMCLR[3:0]: set claim tag bits
Write:
0000: no effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
**ETF lock access register (ETF_LAR)**

Address offset: 0xFB0  
Reset value: 0xxxx x000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**ETF lock status register (ETF_LSR)**

Address offset: 0xFB4  
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:4: Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: reset claim tag bits

Write:
0000: no effect  
xxx1: Clears bit 0..  
x11x: Clears bit 1  
x1xx: Clears bit 2.  
1xxx: Clears bit 3.

Read: Returns current value of claim tag.

**ETF register access enable**

Enables write access to some ETF registers by processor cores (debuggers do not need to unlock the component)

0xC5ACCE55: write access enabled  
Other values: write access disabled

**ETF lock status register (ETF_LSR)**

Address offset: 0xFB4  
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOCKTYPE</th>
<th>LOCKGRANT</th>
<th>LOCKEXIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>f</td>
</tr>
</tbody>
</table>
Bits 31:33 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: size of the ETF_LAR register
- 0: 32-bit
- 1: other size

Bit 1 **LOCKGRANT**: current status of lock
- This bit always returns zero when read by an external debugger.
- 0: write access permitted
- 1: write access blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: existence of lock control mechanism
- The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.
- 0: no lock control mechanism
- 1: lock control mechanism implemented

### ETF authentication status register (ETF_AUTHSTAT)

Address offset: 0xFB8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>31:8</th>
<th>6:3</th>
<th>5:2</th>
<th>4:1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Reserved</td>
<td>SNID[1:0]</td>
<td>SID[1:0]</td>
</tr>
</tbody>
</table>

- Bits 31:8 Reserved, must be kept at reset value.
- Bits 7:6 **SNID[1:0]**: security level for secure non-invasive debug
  - 0x0: not implemented
- Bits 5:4 **SID[1:0]**: security level for secure invasive debug
  - 0x0: not implemented
- Bits 3:2 **NSNID[1:0]**: security level for non-secure non-invasive debug
  - 0x0: not implemented
- Bits 1:0 **NSID[1:0]**: security level for non-secure invasive debug
  - 0x0: not implemented
**ETF device configuration register (ETF_DEVID)**

Address offset: 0xFC8  
Reset value: 0x0000 01C0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MEMWIDTH[2:0]</td>
<td>CONFIGTYP[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **MEMWIDTH[2:0]**: memory interface data bus width  
0x3: 64 bits (corresponds to 32-bit ATB data)

Bits 7:6 **CONFIGTYP[1:0]**: configuration type of component (ETB, ETR or ETF)  
0x2: ETF

Bit 5 **CLKSCHEM**: RAM clocking scheme (synchronous or asynchronous)  
0: synchronous

Bits 4:0 **ATBINPORTCNT[4:0]**: number/type of ATB input port multiplexing  
0x0: none

**ETF device type identifier register (ETF_DEVTYPE)**

Address offset: 0xFCC  
Reset value: 0x0000 0032

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SUBTYPE[3:0]</td>
<td>MAJORTYPE[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification  
0x3: Captures trace data from the ATB slave interface into RAM that can be drained through the ATB master interface

Bits 3:0 **MAJORTYPE[3:0]**: Major classification  
0x2: Component is a trace link because it has an ATB master interface through which trace data can be drained out in Hardware FIFO mode.
ETF CoreSight peripheral identity register 4 (ETF_PIDR4)

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4KCOUNT[3:0]</td>
<td>JEP106CON[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **4KCOUNT[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0  **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ETF CoreSight peripheral identity register 0 (ETF_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0061

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PARTNUM[7:0]**: part number field, bits [7:0]
0x61: ETF part number
### ETF CoreSight peripheral identity register 1 (ETF_PIDR1)

Address offset: 0xFE4  
Reset value: 0x0000 00B9

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **JEP106ID[3:0]:** JEP106 identity code field, bits [3:0]  
0xB: Arm® JEDEC code

Bits 3:0  **PARTNUM[11:8]:** part number field, bits [11:8]  
0x9: ETF part number

### ETF CoreSight peripheral identity register 2 (ETF_PIDR2)

Address offset: 0xFE8  
Reset value: 0x0000 001F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVISION[3:0]:** component revision number  
0x1: r0p1

Bit 3  **JEDEC:** JEDEC assigned value  
1: designer ID specified by JEDEC

Bits 2:0  **JEP106ID[6:4]:** JEP106 identity code field, bits [6:4]  
0x3: Arm® JEDEC code
ETF CoreSight peripheral identity register 3 (ETF_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>REVAND[3:0]</td>
<td>CMOD[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVAND[3:0]**: metal fix version
- 0x0: no metal fix

Bits 3:0  **CMOD[3:0]**: customer modified
- 0x0: no customer modification

ETF CoreSight component identity register 0 (ETF_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PREAMBLE[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[7:0]**: component ID field, bits [7:0]
- 0x0D: common ID value
ETF CoreSight component identity register 1 (ETF_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 0090

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  CLASS[3:0]: component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0  PREAMBLE[11:8]: component ID field, bits [11:8]
0x0: common ID value

ETF CoreSight component identity register 2 (ETF_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  PREAMBLE[19:12]: component ID field, bits [23:16]
0x05: common ID value

ETF CoreSight component identity register 3 (ETF_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  PREAMBLE[27:20]: component ID field, bits [31:24]
0x0B: common ID value
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[27:20]**: component ID field, bits [31:24]

0xB1: common ID value

ETF register map and reset values

Table 528. ETF register map and reset values

| Offset | Register name  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x004  | ETF_RSZ        | RSZ[30:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | Reserved       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00C  | ETF_STS       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x010  | ETF_RRD       | RRD[31:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x014  | ETF_RRP       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x018  | ETF_RWP       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x01C  | ETF_TRG       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x020  | ETF_CTL       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x024  | ETF_RWD       | RWD[31:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x028  | ETF_MODE      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x02C  | ETF_LBUFLVL    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x030  | ETF_CBUFLVL    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x034  | ETF_BUFWM      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x038 to 0x2FC | Reserved       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x300  | ETF_FFSR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

ETF: ETF (ETernal FFl)
| Offset   | Register name          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x304   | ETF_FFCR               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x308   | ETF_PSCR               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 1  | 0  | 1  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x30C   | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x30C   | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFA0   | ETFCLAIMSET            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFA4   | ETFCLAIMCLR            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFA8   | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFA8   | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFB0   | ETF_LAR                | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
|         | Reset value            | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0xFB4   | ETF_LSR                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFB8   | ETF_AUTHSTAT           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xFC0   | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFC0   | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFC8   | ETF_DEVID              | 0  | 0  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|         | Reset value            | 0  | 0  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xFC8   | ETF_DEVTYPE            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 0  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
64.5.5 Trace port interface unit (TPIU)

The TPIU is a CoreSight component that formats the trace stream and outputs it on the external trace port signals. The TPIU has a single ATB slave port for incoming trace data. The trace port is a synchronous parallel port, comprising a clock output, TRACECK, and four data outputs, TRACED(3:0). The trace port width is programmable in the range 1 to 4. Using a smaller port width reduces the number of test points/connector pins needed, and frees up I/Os for other purposes. However it restricts the bandwidth of the trace port and hence the quantity of trace information that can be output in real time. The TRACECK output must be enabled by setting the TRACECLKEN bit in the DBGMCU_CR register.
before trace is sent to the TPIU. Furthermore, the TRACECK frequency can be programmed in the RCC.

For more information on the trace port interface CoreSight component, refer to the Arm® CoreSight SoC-400 technical reference manual [2].

**TPIU registers**

**TPIU supported port size register (TPIU_SUPPSIZE)**

Address offset: 0x000
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>PORTSIZE[31:0]: supported trace port sizes, from 1 to 32 pins. Bit n-1 when set indicates that port size n is supported. 0x0000 000F: port sizes 1 to 4 supported</th>
</tr>
</thead>
</table>

**TPIU current port size register (TPIU_CURPSIZE)**

Address offset: 0x004
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>PORTSIZE[31:0]: current trace port size Bit n-1 when set indicates that the current port size is n pins. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behavior may result. This register must only be modified when the formatter is stopped.</th>
</tr>
</thead>
</table>
### TPIU supported trigger modes register (TPIU_SUPTRGM)

Address offset: 0x100
Reset value: 0x0000 011F

<table>
<thead>
<tr>
<th>Bit 31:18</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 17</td>
<td><strong>TRGRUN</strong>: trigger running</td>
</tr>
<tr>
<td></td>
<td>0: Trigger has not occurred or counter is at 0.</td>
</tr>
<tr>
<td></td>
<td>1: Trigger has occurred and counter is not at 0.</td>
</tr>
<tr>
<td>Bit 16</td>
<td><strong>TRIGD</strong>: triggered</td>
</tr>
<tr>
<td></td>
<td>0: Trigger has not occurred.</td>
</tr>
<tr>
<td></td>
<td>1: Trigger has occurred and counter has reached 0.</td>
</tr>
<tr>
<td>Bits 15:9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 8</td>
<td><strong>TCOUNT8</strong>: 8-bit counter register</td>
</tr>
<tr>
<td></td>
<td>1: implemented</td>
</tr>
<tr>
<td>Bits 7:5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 4</td>
<td><strong>MULT64K</strong>: multiplying the trigger counter by 65536 support</td>
</tr>
<tr>
<td></td>
<td>1: supported</td>
</tr>
<tr>
<td>Bit 3</td>
<td><strong>MULT256</strong>: multiplying the trigger counter by 256 support</td>
</tr>
<tr>
<td></td>
<td>1: supported</td>
</tr>
<tr>
<td>Bit 2</td>
<td><strong>MULT16</strong>: multiplying the trigger counter by 16 support</td>
</tr>
<tr>
<td></td>
<td>1: supported</td>
</tr>
<tr>
<td>Bit 1</td>
<td><strong>MULT4</strong>: multiplying the trigger counter by 4 support</td>
</tr>
<tr>
<td></td>
<td>1: supported</td>
</tr>
<tr>
<td>Bit 0</td>
<td><strong>MULT2</strong>: multiplying the trigger counter by 2 support</td>
</tr>
<tr>
<td></td>
<td>1: supported</td>
</tr>
</tbody>
</table>

---

Note: The table above illustrates the format and functionality of the TPIU_SUPPORTED_TRIGGER_MODES register, detailing bit usage and their respective meanings and supported operations.
### TPIU trigger counter value register (TPIU_TRGCNT)

Address offset: 0x104  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **TRIGCOUNT[7:0]**: enable trigger delay indication

This field enables delaying the indication of triggers to any external connected trace capture or storage devices. This counter is only 8-bit wide and is intended to be used only with the counter multipliers in the trigger multiplier register, 0x108. When a trigger is started, this value, in combination with the multiplier, is the number of words before the trigger is indicated. When the trigger counter reaches 0, the value written here is reloaded. Writing to this register causes the trigger counter value to reset but does not reset any values on the multiplier. Reading this register returns the preset value, not the current count.

### TPIU trigger multiplier register (TPIU_TRGMULT)

Address offset: 0x108  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:5  Reserved, must be kept at reset value.

Bit 4  **MULT64K**: multiply the trigger counter by 65536

0: disabled  
1: enabled

Bit 3  **MULT256**: multiply the trigger counter by 256

0: disabled  
1: enabled
Bit 2  **MULT16**: multiply the trigger counter by 16  
0: disabled  
1: enabled  

Bit 1  **MULT4**: multiply the trigger counter by 4  
0: disabled  
1: enabled  

Bit 0  **MULT2**: multiply the trigger counter by 2  
0: disabled  
1: enabled

**TPIU supported test patterns/modes register (TPIU_SUPTPM)**

Address offset: 0x200  
Reset value: 0x0003 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:18  Reserved, must be kept at reset value.  

Bit 17  **PCONTEN**: support of continuous mode  
1: supported  

Bit 16  **PTIMEEN**: support of timed mode  
1: supported  

Bits 15:4  Reserved, must be kept at reset value.  

Bit 3  **PATF0**: support of FF/00 pattern  
This bit indicates whether the FF/00 pattern is supported as output over the trace port.  
1: supported  

Bit 2  **PATA5**: support of AA/55 pattern  
This bit indicates whether the AA/55 pattern is supported as output over the trace port.  
1: supported  

Bit 1  **PATW0**: support of walking 0’s pattern  
This bit indicates whether the walking 0’s pattern is supported as output over the trace port.  
1: supported  

Bit 0  **PATW1**: support of walking 1’s pattern  
This bit indicates whether the walking 1’s pattern is supported as output over the trace port.  
1: supported
## TPIU current test pattern/mode register (TPIU_CURTPM)

Address offset: 0x204  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:18</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 17 | **PCONTEN**: continuous mode enable  
0: disabled  
1: enabled |
| Bit 16 | **PTIMEEN**: timed mode enable  
0: disabled  
1: enabled |
| Bits 15:4 | Reserved, must be kept at reset value. |
| Bit 3 | **PATF0**: FF/00 pattern enable  
This bit indicates whether the FF/00 pattern is enabled as output over the trace port  
0: disabled  
1: enabled |
| Bit 2 | **PATA5**: AA/55 pattern enable  
This bit indicates whether the AA/55 pattern is enabled as output over the trace port  
0: disabled  
1: enabled |
| Bit 1 | **PATW0**: walking 0's pattern enable  
This bit indicates whether the walking 0's pattern is enabled as output over the trace port  
0: disabled  
1: enabled |
| Bit 0 | **PATW1**: walking 1's pattern enable  
This bit indicates whether the walking 1's pattern is enabled as output over the trace port  
0: disabled  
1: enabled |
### TPIU test pattern repeat counter register (TPIU_TPRCR)

Address offset: 0x208  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x208</td>
<td>TPIU_TPRCR</td>
<td>TPIU test pattern repeat counter register</td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>PATTCOUNT[7:0]</td>
<td>PATTCOUNT[7:0]: number of TRACECLKIN cycles</td>
</tr>
</tbody>
</table>
|                |                  | This field provides a 8-bit counter value to indicate the number of TRACECLKIN cycles for which a pattern runs before it switches to the next pattern.

#### Bits 31:8
Reserved, must be kept at reset value.

#### Bits 7:0
**PATTCOUNT[7:0]: number of TRACECLKIN cycles**
This field provides a 8-bit counter value to indicate the number of TRACECLKIN cycles for which a pattern runs before it switches to the next pattern.

### TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300  
Reset value: 0x0000 0002

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x300</td>
<td>TPIU_FFSR</td>
<td>TPIU formatter and flush status register</td>
</tr>
</tbody>
</table>
| 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 | TCPRESENT | TCPRESENT: TRACECTL output pin availability  
This bit indicates whether the optional TRACECTL output pin is available for use.  
0: TRACECTL pin is not present in this device.
|                | FTSTOPPED        | FTSTOPPED: formatter stopped  
The formatter has received a stop request signal and all trace data and post-amble are sent.  
Any additional trace data on the ATB interface is ignored.  
0: Formatter has not stopped.  
1: Formatter has stopped.
|                | FLINPROG         | FLINPROG: flush in progress  
This bit indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.  
0: no flush in progress  
1: flush in progress.

#### Bits 31:3
Reserved, must be kept at reset value.

#### Bit 2
**TCPRESENT**: TRACECTL output pin availability  
This bit indicates whether the optional TRACECTL output pin is available for use.  
0: TRACECTL pin is not present in this device.

#### Bit 1
**FTSTOPPED**: formatter stopped  
The formatter has received a stop request signal and all trace data and post-amble are sent.  
Any additional trace data on the ATB interface is ignored.  
0: Formatter has not stopped.  
1: Formatter has stopped.

#### Bit 0
**FLINPROG**: flush in progress  
This bit indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.  
0: no flush in progress  
1: flush in progress.
## TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>Res.</td>
<td>STOPTRIG</td>
<td>STOPFL</td>
<td>TRIGFL</td>
<td>TRIGEVT</td>
<td>TRIGIN</td>
<td>Res.</td>
<td>FONMAN</td>
<td>FONTRIG</td>
<td>FONFLIN</td>
<td>Res.</td>
<td>Res.</td>
<td>ENFCONT</td>
<td>ENFTC</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **STOPTRIG**: stop on trigger event
- 0: no effect
- 1: Stops formatter when a trigger event occurs.

Bit 12 **STOPFL**: stop on flush
- 0: no effect
- 1: Stops formatter when flush is completed.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **TRIGFL**: trigger on flush
- 0: no effect
- 1: Indicates a trigger in the trace stream when flush is completed.

Bit 9 **TRIGEVT**: trigger on trigger event
- 0: no effect
- 1: Indicates a trigger in the trace stream when trigger event occurs.

Bit 8 **TRIGIN**: trigger on trigger in
- 0: No effect
- 1: Indicates a trigger in the trace stream when the TRIGIN input from the system CTI is asserted.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FONMAN**: generate a manual flush
- 0: no effect
- 1: Flushes the trace.  
  This bit is cleared automatically when the flush completes.

Bit 5 **FONTRIG**: flush on trigger event
A trigger event occurs when the trigger counter reaches 0, or, if the trigger counter is 0, when the TRIGIN input from the system CTI is high.
- 0: no effect
- 1: Flushes the trace if a trigger event occurs.

Bit 4 **FONFLIN**: flush on flush in
- 0: no effect
- 1: Flushes the trace if the FLUSHIN input from the system CTI is asserted.
Bits 3:2  Reserved, must be kept at reset value.

Bit 1  ENFCONT: enable continuous formatting
   0: continuous formatting disabled
   1: continuous formatting enabled

Bit 0  ENFTC: enable the embedding of triggers in formatted trace
   0: formatting disabled
   1: formatting enabled

**TPIU formatter synchronization counter register (TPIU_FSCR)**

Address offset: 0x400
Reset value: 0x0000 0040

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**CYCCOUNT[11:0]**

Bits 31:12  Reserved, must be kept at reset value.

Bits 11:0  **CYCCOUNT[11:0]: effective use of TPAs enable**

This field enables the effective use of different-sized TPAs without wasting large amounts of storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word sync frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

**TPIU claim tag set register (TPIU_CLAIMSET)**

Address offset: 0xFA0
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**CLAIMSET[3:0]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

** rw rw rw rw **
Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **CLAIMSET[3:0]**: set claim tag bits
Write:
- 0000: no effect.
- xxx1: Sets bit 0.
- xx1x: Sets bit 1.
- x1xx: Sets bit 2.
- 1xxx: Sets bit 3.
Read:
- 0xF: Indicates there are four bits in claim tag.

**TPIU claim tag clear register (TPIU_CLAIMCLR)**
Address offset: 0xFA4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **CLAIMCLR[3:0]**: reset claim tag bits
Write:
- 0000: no effect
- xxx1: Clears bit 0.
- xx1x: Clears bit 1.
- x1xx: Clears bit 2.
- 1xxx: Clears bit 3.
Read: Returns current value of claim tag.

**TPIU lock access register (TPIU_LAR)**
Address offset: 0xFB0
Reset value: 0xFFFF XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>
Bits 31:0 **ACCESS_W[31:0]**: TPIU register access enable

This register enables write access to some TPIU registers by processor cores (debuggers do not need to unlock the component)

- 0xC5ACCE55: write access enabled
- Other values: write access disabled

**TPIU lock status register (TPIU_LSR)**

Address offset: 0xFB4

Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>ACCESS_W[31:0]</td>
</tr>
</tbody>
</table>

Bits 31:3 Reserved, must be kept at reset value.

Bit 2  **LOCKTYPE**: size of the TPIU_LAR register
0: 32-bit

Bit 1  **LOCKGRANT**: current status of lock
This bit always returns zero when read by an external debugger.
0: write access permitted
1: write access blocked. Only read access is permitted.

Bit 0  **LOCKEXIST**: existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.
0: no lock control mechanism
1: lock control mechanism implemented

**TPIU authentication status register (TPIU_AUTHSTAT)**

Address offset: 0xFB8

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>ACCESS_W[31:0]</td>
</tr>
</tbody>
</table>

Bits 31:3 Reserved, must be kept at reset value.

Bit 2  **LOCKTYPE**: size of the TPIU_LAR register
0: 32-bit

Bit 1  **LOCKGRANT**: current status of lock
This bit always returns zero when read by an external debugger.
0: write access permitted
1: write access blocked. Only read access is permitted.

Bit 0  **LOCKEXIST**: existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.
0: no lock control mechanism
1: lock control mechanism implemented
**TPIU device configuration register (TPIU_DEVID)**

Address offset: 0xFC8  
Reset value: 0x0000 00A0

| Bit 31:11 Reserved, must be kept at reset value. |
| Bit 11 **SWOUARTNRZ**: support of SWO UART or NRZ  
This bit indicates whether serial wire output, UART or NRZ, is supported.  
0: not supported |
| Bit 10 **SWOMAN**: support of SWO Manchester format  
This bit indicates whether serial wire output, Manchester encoded format, is supported.  
0: not supported |
| Bit 9 **TCLKDATA**: support of trace clock plus data  
0: not supported |
| Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2  
0x2: FIFO size = 4 (16 bytes) |
| Bit 5 **CLKRELAT**: ATB clock and TRACECLKIN relation  
Indicates the relationship between the ATB clock and TRACECLKIN (synchronous or asynchronous)  
1: asynchronous |
| Bits 4:0 **MUXNUM[4:0]**: number/type of ATB input port multiplexing  
0x0: none |
**TPIU device type identifier register (TPIU_DEVTYPE)**

Address offset: 0xFCC

Reset value: 0x0000 0011

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **SUBTYPE[3:0]**: sub-classification

- 0x1: trace port component

Bits 3:0  **MAJORTYPE[3:0]**: major classification

- 0x1: trace sink component

**TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)**

Address offset: 0xFD0

Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **4KCOUNT[3:0]**: register file size

- 0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0  **JEP106CON[3:0]**: JEP106 continuation code

- 0x4: Arm® JEDEC code
### TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0  
Reset value: 0x0000 0012

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PARTNUM[7:0]**: part number field, bits [7:0]  
0x12: TPIU part number

### TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4  
Reset value: 0x0000 00B9

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]  
0xB: Arm® JEDEC code

Bits 3:0  **PARTNUM[11:8]**: part number field, bits [11:8]  
0x9: TPIU part number
### TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)

**Address offset:** 0xFE8  
**Reset value:** 0x0000 004B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Bits 31:8:** Reserved, must be kept at reset value.
- **Bits 7:4** `REVISION[3:0]`: component revision number
  - 0x4: r0p5
- **Bit 3** `JEDEC`: JEDEC assigned value
  - 1: designer ID specified by JEDEC
  - 0x3: Arm® JEDEC code

### TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)

**Address offset:** 0xFEC  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Bits 31:8:** Reserved, must be kept at reset value.
- **Bits 7:4** `REVAND[3:0]`: metal fix version
  - 0x0: no metal fix
- **Bits 3:0** `CMOD[3:0]`: customer modified
  - 0x0: no customer modifications
## TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0  
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[7:0]:** component ID field, bits [7:0]  
0x0D: common ID value

## TPIU CoreSight component identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4  
Reset value: 0x0000 0090

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **CLASS[3:0]:** component ID field, bits [15:12] - component class  
0x9: CoreSight component

Bits 3:0  **PREAMBLE[11:8]:** component ID field, bits [11:8]  
0x0: common ID value
TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  PREAMBLE[19:12]: component ID field, bits [23:16]
          0x05: common ID value

TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  PREAMBLE[27:20]: component ID field, bits [31:24]
          0xB1: common ID value

TPIU register map and reset values

| Offset | Register name  | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x000  | TPIU_SUPPSIZE  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x004  | TPIU_CURPSIZE  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x008 to 0x0FC | Reserved     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Table 529. TPIU register map and reset values
### Table 529. TPIU register map and reset values (continued)

| Offset   | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x100    | TPIU_SUPTRGM  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |     | Reset value | 0  | 0  | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x104    | TPIU_TRGCNT  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |     | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x108    | TPIU_TRGMULT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |     | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10C to 0x1FC | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x200    | TPIU_SUPTPM  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |     | Reset value | 1  | 1  | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x204    | TPIU_CURTPM  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |     | Reset value | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x208    | TPIU_TPRCR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |     | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x20C to 0x2FC | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x300    | TPIU_FFSR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |     | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x304    | TPIU_FFCR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |     | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x308 to 0x39C | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x400    | TPIU_FSCR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |     | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x404 to 0x49C | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFA0    | TPIU_CLAIMSET |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |     | Reset value | 1  | 1  | 1  | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
### Table 529. TPIU register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 0xFA4  | TPIU_CLAIMCLR |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value   |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0xFA8  | Reserved      |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0xFA4  | TPIU_LAR      |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value   | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      |
| 0xFB4  | TPIU_LSR      |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value   |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0xFB8  | TPIU_AUTHSTAT |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value   |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0xFC8  | TPIU_DEVID    |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value   | 0      | 0      | 0      | 1      | 0      | 1      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| 0xFD0  | TPIU_PIDR4    |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value   | 0      | 0      | 0      | 0      | 1      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| 0xFE0  | TPIU_PIDR0    |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value   | 0      | 0      | 0      | 1      | 0      | 0      | 0      | 1      | 0      | 0      | 0      | 1      | 0      | 0      | 0      | 1      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| 0xFE4  | TPIU_PIDR1    |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        | Reset value   | 1      | 0      | 1      | 1      | 1      | 0      | 0      | 0      | 1      | 0      | 0      | 0      | 0      | 1      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
64.5.6 Serial-wire output (SWO)

The SWO is a CoreSight component that formats the trace stream from the processor ITM and outputs it on the single wire TRACESWO output.

Compared to the TPIU, the SWO contains:

- no formatter
- no pattern generator
- an 8-bit ATB input
- no synchronous trace output, that is, no TRACEDATA or TRACECLK pins
- no support for flush
- no support for triggering

The SWO output supports Manchester encoded and UART NRZ formats.

For more information about the serial wire output CoreSight component, refer to the Arm® CoreSight Components Technical Reference Manual [4].
SWO registers

SWO current output divisor register (SWO_CODR)

Address offset: 0x010
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:13  Reserved, must be kept at reset value.

Bits 12:0  **PRESCALER[12:0]**: SWO baud rate scaling

The baud rate is the trace clock frequency divided by (PRESCALER - 1). The baud rate changes instantly, so it is recommended to stop the trace source and wait until the port is idle before writing to this register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:2  Reserved, must be kept at reset value.

Bits 1:0  **PPROT[1:0]**: pin protocol

- 0x0: reserved
- 0x1: Manchester
- 0x2: NRZ
- 0x3: reserved
**SWO formatter and flush status register (SWO_FFSR)**

Address offset: 0x300  
Reset value: 0x0000 0008

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FTNONSTOP</td>
<td>TCPRESENT</td>
<td>FTSTOPPED</td>
<td>FLINPROG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.

Bit 3  **FTNONSTOP**: change of settings without stopping formatter  
1: Change of settings is allowed with formatter running.

Bit 2  **TCPRESENT**: TRACECTL pin present on SWO  
0: TRACECTL pin not present

Bit 1  **FTSTOPPED**: formatter stopped  
0: formatter running  
The bit always returns 0 as the SWO formatter cannot be stopped in this device.

Bit 0  **FLINPROG**: flush in progress  
0: Flush is not in progress.  
The bit always returns 0 as SWO flushing is not supported in this device.

**SWO claim tag set register (SWO_CLAIMSET)**

Address offset: 0xFA0  
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CLAIMSET[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RM0455 Debug infrastructure

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **CLAIMSET[3:0]**: Sst claim tag bits

Write:
- 0000: no effect
- xxx1: Sets bit 0.
- xx1x: Sets bit 1.
- x1xx: Sets bit 2.
- 1xxx: Sets bit 3.

Read:
- 0xF: Indicates there are four bits in claim tag.

**SWO claim tag clear register (SWOCLAIMCLR)**

Address offset: 0xFA4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CLAIMCLR[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **CLAIMCLR[3:0]**: reset claim tag bits

Write:
- 0000: no effect
- xxx1: Clears bit 0.
- xx1x: Clears bit 1.
- x1xx: Clears bit 2.
- 1xxx: Clears bit 3.

Read: Returns current value of claim tag.

**SWO lock access register (SWO_LAR)**

Address offset: 0xFB0
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

RM0455 Rev 3 2841/2974
Bits 31:0  **ACCESS_W[31:0]**: SWO register write access enable
This register enables write access to some SWO registers by processor cores (debuggers do not need to unlock the component)
0xC5ACCE55: write access enabled
Other values: write access disabled

**SWO lock status register (SWO_LSR)**
Address offset: 0xFB4
Reset value: 0x0000 0003

![SWO_LSR Register](image)

Bits 31:3  Reserved, must be kept at reset value.

Bit 2  **LOCKTYPE**: size of the SWO_LAR register
0: 32-bit

Bit 1  **LOCKGRANT**: current status of lock
This bit always returns zero when read by an external debugger.
0: write access permitted
1: write access blocked - only read access is permitted

Bit 0  **LOCKEXIST**: existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.
0: no lock control mechanism
1: lock control mechanism implemented

**SWO authentication status register (SWO_AUTHSTAT)**
Address offset: 0xFB8
Reset value: 0x0000 0000

![SWO_AUTHSTAT Register](image)
RM0455 Debug infrastructure

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:6  **SNID[1:0]**: security level for secure non-invasive debug  
0x0: not implemented

Bits 5:4  **SID[1:0]**: security level for secure invasive debug  
0x0: not implemented

Bits 3:2  **NSNID[1:0]**: security level for non-secure non-invasive debug  
0x0: not implemented

Bits 1:0  **NSID[1:0]**: security level for non-secure invasive debug  
0x0: not implemented

**SWO device configuration register (SWO_DEVID)**

Address offset: 0xFC8  
Reset value: 0x0000 0EA0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:11  Reserved, must be kept at reset value.

Bit 11  **SWOUARTNRZ**: SWO UART or NRZ support  
This bit indicates whether serial wire output, UART or NRZ, is supported.  
1: supported

Bit 10  **SWOMAN**: SWO Manchester format support  
This bit indicates whether serial wire output, Manchester encoded format, is supported.  
1: supported

Bit 9  **TCLKDATA**: trace clock plus data support  
This bit indicates whether trace clock plus data is supported  
1: supported

Bits 8:6  **FIFOSIZE[2:0]**: FIFO size in powers of 2  
0x2: FIFO size = 4 (16 bytes)

Bit 5  **CLKRELAT**: ATB clock to TRACECLKIN relation  
This bit indicates the relationship between the ATB clock and TRACECLKIN (synchronous or asynchronous)  
1: asynchronous

Bits 4:0  **MUXNUM[4:0]**: number/type of ATB input port multiplexing  
0x0: none
**SWO device type identifier register (SWO_DEVTYPE)**

Address offset: 0xFCC  
Reset value: 0x0000 0011

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----------</td>
<td>--------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
</tbody>
</table>
```

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: sub-classification  
0x1: trace port component

Bits 3:0 **MAJORTYPE[3:0]**: major classification  
0x1: trace sink component

**SWO CoreSight peripheral identity register 4 (SWO_PIDR4)**

Address offset: 0xFD0  
Reset value: 0x0000 0004

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----------</td>
<td>-------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
</tbody>
</table>
```

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size  
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code  
0x4: Arm® JEDEC code
SWO CoreSight peripheral identity register 0 (SWO_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0014

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PARTNUM[7:0]**: part number field, bits [7:0]
0x14: SWO part number

SWO CoreSight peripheral identity register 1 (SWO_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Parts 31:8  Reserved, must be kept at reset value.

Bits 7:4  **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0  **PARTNUM[11:8]**: part number field, bits [11:8]
0x9: SWO part number
### SWO CoreSight peripheral identity register 2 (SWO_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 001B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
- 0x1: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
- 1: designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
- 0x3: Arm® JEDEC code

### SWO CoreSight peripheral identity register 3 (SWO_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
- 0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
- 0x0: no customer modifications
### SWO CoreSight component identity register 0 (SWO_CIDR0)

Address offset: 0xFF0  
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.  

Bits 7:0  **PREAMBLE[7:0]:** component ID field, bits [7:0]  
0xD: common ID value  

### SWO CoreSight component identity register 1 (SWO_CIDR1)

Address offset: 0xFF4  
Reset value: 0x0000 0090

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.  

Bits 7:4  **CLASS[3:0]:** component ID field, bits [15:12] - component class  
0x9: CoreSight component  

Bits 3:0  **PREAMBLE[11:8]:** component ID field, bits [11:8]  
0x0: common ID value
### SWO CoreSight component identity register 2 (SWO_CIDR2)

Address offset: 0xFF8  
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10</td>
<td>SWO_CDDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRESCALER[12:0]</td>
<td>0000000000000</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[19:12]**: component ID field, bits [23:16]
0x05: common ID value

### SWO CoreSight component identity register 3 (SWO_CIDR3)

Address offset: 0xFFC  
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x14</td>
<td>SWO_SPPR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PPROT[1:0]</td>
<td>01</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[27:20]**: component ID field, bits [31:24]
0xB1: common ID value

### SWO register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x010</td>
<td>SWO_CDDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRESCALER[12:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>0x014</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0EC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0F0</td>
<td>SWO_SPPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 530. SWO register map and reset values
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF0 to 0x2FC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x300</td>
<td>SWO_FFSR</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x304 to 0x3FC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFA0</td>
<td>SWO_CLAIMSET</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFA4</td>
<td>SWO_CLAIMCLR</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFA8 to 0xFBAC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFB0</td>
<td>SWO_LAR</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACCESS_W[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFB4</td>
<td>SWO_LSR</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFB8</td>
<td>SWO_AUTHSTAT</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFBCC to 0xFC4</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFC8</td>
<td>SWO_DEVID</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFCCC</td>
<td>SWO_DEVTYPE</td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Table 530. SWO register map and reset values (continued)

| Offset  | Register name | Offset | Reset value | S.M.D.
|---------|---------------|--------|-------------|--------
| 0xFD0   | SWO_PIDR4     | 31-30 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 |          |
| 0xFD4 to 0xFD8 | Reserved |        |             | Reserved |
| 0xFE0   | SWO_PIDR0     | 31-30 | 0 0 0 1 0 1 0 0 0 0 | PARTNUM[7:0] |
| 0xFE4   | SWO_PIDR1     | 31-30 | 1 0 1 1 1 0 0 1 | JEP[106][D17:0] |
| 0xFE8   | SWO_PIDR2     | 31-30 | 0 0 0 1 1 0 1 1 | REV[106][D11:0] |
| 0xFE8 to 0xFEC | Reserved |        |             | Reserved |
| 0xFE8   | SWO_PIDR3     | 31-30 | 0 0 0 0 0 0 0 0 0 0 0 | REVAND[3:0] |
| 0xFF0   | SWO_CIDR0     | 31-30 | 0 0 0 0 1 1 0 1 | CMOD[3:0] |
| 0xFF4   | SWO_CIDR1     | 31-30 | 1 0 0 1 0 0 0 0 0 0 | CLASS[3:0] |
| 0xFF8   | SWO_CIDR2     | 31-30 | 0 0 0 0 0 0 0 0 0 0 0 | PREAMBLE[19:12] |
| 0xFFC   | SWO_CIDR3     | 31-30 | 1 0 1 1 0 0 0 0 0 0 | PREAMBLE[27:20] |
64.5.7 Microcontroller debug unit (DBGMCU)

The DBGMCU component contains a number of registers that control the power and clock behavior in debug mode. Specifically it allows the debugger, or debug software, to:

- maintain the clock and power to the processor core when in low-power modes (Sleep, Stop or Standby)
- maintain the clock and power to the system debug and trace components when in low-power modes
- stop the clock to certain peripherals (CAN, SMBUS timeout, watchdogs, timers, RTC) when the processor core is stopped in debug mode. For timers having complementary outputs, the outputs are disabled (as if the MOE bit was reset) for safety purposes when the counter is stopped (TIM1/8/15/16/17 = 1 in DBGMCU_APB2FZ1).

The DBGMCU registers are not reset by a system reset, only by a power-on reset. They are accessible to the debugger via the APB-D bus at base address 0xE00E1000. They are also accessible by software at base address 0x5C001000.

Note: the DBGMCU is not a standard CoreSight component. Therefore, it does not appear in the system ROM table.

DBGMCU registers

DBGMCU identity code register (DBGMCU_IDC)

Address offset: 0x000
Reset value: 0x1000 6480

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REV_ID[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEV_ID[11:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16  **REV_ID[15:0]**: revision
- 0x1000 = revision A
- 0x1001 = revision Z

Bits 15:12  Reserved, must be kept at reset value.

Bits 11:0  **DEV_ID[11:0]**: device ID
- 0x480: STM32H7A3/7B3/7B0
### DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x004

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRGOEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

- **Bit 28 TRGOEN**: external trigger output enable
  - This bit controls the direction of the bi-directional trigger pin, TRGIO.
  - 0: input - TRGIO is connected to TRGIN
  - 1: output - TRGIO is connected to TRGOUT

Bits 27:23 Reserved, must be kept at reset value.

- **Bit 22 SRDDBGCKEN**: SmartRun domain debug clock enable
  - This bit allows the debug components in the SmartRun domain (excluding the DAPCLK domain) to be switched off if they are not needed.
  - 0: disabled - SmartRun domain debug components are disabled and their clocks gated.
  - 1: enabled - SmartRun domain debug components are clocked whenever the corresponding domain clock (rcc_fclk_srd) is active.

- **Bit 21 CDDBGCKEN**: CPU domain debug clock enable
  - This bit allows the debug components in the CPU domain (excluding those in the processor core) to be switched off if they are not needed.
  - 0: disabled - CPU domain debug components are disabled and their clocks gated.
  - 1: enabled - CPU domain debug components are clocked whenever the corresponding domain clock (rcc_aclk) is active.

- **Bit 20 TRACECLKEN**: trace port clock enable
  - This bit enables the trace port clock, TRACECLK.
  - 0: TRACECLK disabled
  - 1: TRACECLK enabled

Bits 19:9 Reserved, must be kept at reset value.
Bit 8  **DBGSTBY_SRD**: debug in SmartRun domain Standby mode  
0: normal operation - all clocks are disabled and the domain powered down automatically in Standby mode\(^{(1)}\).  
1: automatic clock stop/power-down disabled - all active clocks and oscillators continue to run during Standby mode, and the SmartRun domain supply is maintained, allowing full debug capability. On exit from Standby mode, a system reset is performed.

Bit 7  **DBGSTOP_SRD**: debug in SmartRun domain Stop mode  
0: normal operation - domain clocks are disabled automatically in Stop mode\(^{(2)}\).  
1: automatic clock stop/power-down disabled - all active clocks and oscillators continue to run during Stop mode. On exit from Stop mode, the clock settings is set to the Stop mode exit state.

Bits 6:3  Reserved, must be kept at reset value.

Bit 2  **DBGSTBY_CD**: CPU domain debug in Standby mode  
0: normal operation - all clocks are disabled and the CPU domain is powered down automatically in Standby mode.  
1: automatic clock stop/power-down disabled - all active clocks and oscillators continue to run during Standby mode, and the CPU domain supply is maintained, allowing full debug capability. On exit from Standby mode, a domain reset is performed.

Bit 1  **DBGSTOP_CD**: CPU domain debug in Stop mode  
0: normal operation - all clocks are disabled automatically in Stop mode.  
1: automatic clock stop disabled - all active clocks and oscillators continue to run during Stop mode, allowing full debug capability. On exit from Stop mode, the clock settings is set to the Stop mode exit state.

Bit 0  **DBGSLEEP_CD**: CPU domain debug in Sleep mode  
0: normal operation - processor clock is stopped automatically in Sleep mode.  
1: automatic clock stop disabled - processor clock continues to run, allowing full debug capability.

---

1. If CDBGPWRUPREQ in the debug port Control/Stat (DP_CTRL/STAT) register is asserted, the SmartRun domain remains powered up and the DAPCLK active in Standby mode, even if the DBGSTBY_SRD bit is reset. However the remaining SmartRun domain clocks are switched off.

2. If CDBGPWRUPREQ in the debug port Control/Stat register is asserted, the SmartRun domain remains powered up and the DAPCLK active in Stop mode, even if the DBGSTOP_SRD bit is reset. However the remaining SmartRun domain clocks are switched off.

**DBGMCU APB3 peripheral freeze register (DBGMCU_APB3FZ1)**  
Address offset: 0x034  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

** rw**
**DBGMCU APB1L peripheral freeze register (DBGMCU_APB1LFZ1)**

Address offset: 0x03C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:24** Reserved, must be kept at reset value.

**Bit 23** I2C3: I2C3 SMBUS timeout stop in debug

- 0: normal operation - I2C3 SMBUS timeout continues to operate while the Cortex-M7 core is in debug mode.
- 1: stop in debug - I2C3 SMBUS timeout is frozen while Cortex-M7 is in debug mode.

**Bit 22** I2C2: I2C2 SMBUS timeout stop in debug

- 0: normal operation - I2C2 SMBUS timeout continues to operate while the Cortex-M7 core is in debug mode.
- 1: stop in debug - I2C2 SMBUS timeout is frozen while Cortex-M7 is in debug mode.

**Bit 21** I2C1: I2C1 SMBUS timeout stop in debug

- 0: normal operation - I2C1 SMBUS timeout continues to operate while the Cortex-M7 core is in debug mode.
- 1: stop in debug - I2C1 SMBUS timeout is frozen while Cortex-M7 is in debug mode.

**Bits 20:10** Reserved, must be kept at reset value.

**Bit 9** LPTIM1: LPTIM1 stop in debug

- 0: normal operation - LPTIM1 continues to operate while the Cortex-M7 core is in debug mode.
- 1: stop in debug - LPTIM1 is frozen while Cortex-M7 is in debug mode.

**Bit 8** TIM14: TIM14 stop in debug

- 0: normal operation - TIM14 continues to operate while the Cortex-M7 core is in debug mode.
- 1: stop in debug - TIM14 is frozen while Cortex-M7 is in debug mode.

**Bit 7** TIM13: TIM13 stop in debug

- 0: normal operation - TIM13 continues to operate while the Cortex-M7 core is in debug mode.
- 1: stop in debug - TIM13 is frozen while Cortex-M7 is in debug mode.

**Bit 6** TIM12: TIM12 stop in debug

- 0: normal operation - TIM12 continues to operate while the Cortex-M7 core is in debug mode.
- 1: stop in debug - TIM12 is frozen while Cortex-M7 is in debug mode.
### DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZ1)

**Address offset:** 0x04C  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
| 18  | TIM17         | 0: normal operation - TIM17 continues to operate while the Cortex-M7 core is in debug mode.  
1: stop in debug - TIM17 is frozen and TIM17 outputs are disabled while Cortex-M7 is in debug mode. |
| 17  | TIM16         | 0: normal operation - TIM16 continues to operate while the Cortex-M7 core is in debug mode.  
1: stop in debug - TIM16 is frozen and TIM16 outputs are disabled while Cortex-M7 is in debug mode. |
| 16  | TIM15         | 0: normal operation - TIM15 continues to operate while the Cortex-M7 core is in debug mode.  
1: stop in debug - TIM15 is frozen and TIM15 outputs are disabled while Cortex-M7 is in debug mode. |
| 15  | Reserved      | Reserved, must be kept at reset value.                                            |
| 14  | Reserved      | Reserved, must be kept at reset value.                                            |
| 13  | Reserved      | Reserved, must be kept at reset value.                                            |
| 12  | Reserved      | Reserved, must be kept at reset value.                                            |
| 11  | Reserved      | Reserved, must be kept at reset value.                                            |
| 10  | Reserved      | Reserved, must be kept at reset value.                                            |
| 9   | Reserved      | Reserved, must be kept at reset value.                                            |
| 8   | Reserved      | Reserved, must be kept at reset value.                                            |
| 7   | Reserved      | Reserved, must be kept at reset value.                                            |
| 6   | Reserved      | Reserved, must be kept at reset value.                                            |
| 5   | Reserved      | Reserved, must be kept at reset value.                                            |
| 4   | Reserved      | Reserved, must be kept at reset value.                                            |
| 3   | Reserved      | Reserved, must be kept at reset value.                                            |
| 2   | Reserved      | Reserved, must be kept at reset value.                                            |
| 1   | TIM8          | 0: normal operation - TIM8 continues to operate while the Cortex-M7 core is in debug mode.  
1: stop in debug - TIM8 is frozen and TIM8 outputs are disabled while Cortex-M7 is in debug mode. |
| 0   | TIM1          | 0: normal operation - TIM1 continues to operate while the Cortex-M7 core is in debug mode.  
1: stop in debug - TIM1 is frozen and TIM1 outputs are disabled while Cortex-M7 is in debug mode. |

**Bit 5 TIM7:** TIM7 stop in debug  
0: normal operation - TIM7 continues to operate while the Cortex-M7 core is in debug mode.  
1: stop in debug - TIM7 is frozen while Cortex-M7 is in debug mode.

**Bit 4 TIM6:** TIM6 stop in debug  
0: normal operation - TIM6 continues to operate while the Cortex-M7 core is in debug mode.  
1: stop in debug - TIM6 is frozen while Cortex-M7 is in debug mode.

**Bit 3 TIM5:** TIM5 stop in debug  
0: normal operation - TIM5 continues to operate while the Cortex-M7 core is in debug mode.  
1: stop in debug - TIM5 is frozen while Cortex-M7 is in debug mode.

**Bit 2 TIM4:** TIM4 stop in debug  
0: normal operation - TIM4 continues to operate while the Cortex-M7 core is in debug mode.  
1: stop in debug - TIM4 is frozen while Cortex-M7 is in debug mode.

**Bit 1 TIM3:** TIM3 stop in debug  
0: normal operation - TIM3 continues to operate while the Cortex-M7 core is in debug mode.  
1: stop in debug - TIM3 is frozen while Cortex-M7 is in debug mode.

**Bit 0 TIM2:** TIM2 stop in debug  
0: normal operation - TIM2 continues to operate while the Cortex-M7 core is in debug mode.  
1: stop in debug - TIM2 is frozen while Cortex-M7 is in debug mode.
Bits 15:2  Reserved, must be kept at reset value.

**Bit 1**  **TIM8**: TIM8 stop in debug
0: normal operation - TIM8 continues to operate while the Cortex-M7 core is in debug mode.
1: stop in debug - TIM8 is frozen and TIM8 outputs are disabled while Cortex-M7 is in debug mode.

**Bit 0**  **TIM1**: TIM1 stop in debug
0: normal operation - TIM1 continues to operate while the Cortex-M7 core is in debug mode.
1: stop in debug - TIM1 is frozen and TIM1 outputs are disabled while Cortex-M7 is in debug mode.

**DBGMCU APB4 peripheral freeze register (DBGMCU_APB4FZ1)**
Address offset: 0x054
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WDG</td>
<td>LSCD</td>
<td>RTC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WDT</td>
<td>LPTIM2</td>
<td>LPTIM1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:19**  Reserved, must be kept at reset value.

**Bit 18**  **WDGLSCD**: LS watchdog for CPU domain stop in debug
0: normal operation - watchdog continues to count while the Cortex-M7 core is in debug mode.
1: stop in debug - watchdog is frozen while Cortex-M7 is in debug mode.

**Bit 17**  Reserved, must be kept at reset value.

**Bit 16**  **RTC**: RTC stop in debug
0: normal operation - RTC continues to operate while the Cortex-M7 core is in debug mode.
1: stop in debug - RTC is frozen while Cortex-M7 is in debug mode.

**Bits 15:11**  Reserved, must be kept at reset value.

**Bit 10**  **LPTIM3**: LPTIM2 stop in debug
0: normal operation - LPTIM2 continues to operate while the Cortex-M7 core is in debug mode.
1: stop in debug - LPTIM2 is frozen while Cortex-M7 is in debug mode.

**Bit 9**  **LPTIM2**: LPTIM2 stop in debug
0: normal operation - LPTIM2 continues to operate while the Cortex-M7 core is in debug mode.
1: stop in debug - LPTIM2 is frozen while Cortex-M7 is in debug mode.

**Bit 8**  Reserved, must be kept at reset value.

**Bit 7**  **I2C4**: I2C4 SMBUS timeout stop in debug
0: normal operation - I2C4 SMBUS timeout continues to operate while the Cortex-M7 core is in debug mode.
1: stop in debug - I2C4 SMBUS timeout is frozen while the Cortex-M7 core is in debug mode.

**Bits 6:0**  Reserved, must be kept at reset value.
The Cortex-M7 subsystem features the following CoreSight components:

- ROM tables
- System control space (SCS)
- Breakpoint unit (FPB)
- Data watchpoint and trace unit (DWT)
- Instrumentation trace macrocell (ITM)
- Embedded trace macrocell (ETM)
- Cross trigger interface (CTI)

These components are accessible by the debugger via the Cortex-M7 AHB-AP and its associated AHBD bus.
64.6.1 Cortex-M7 ROM tables

The ROM tables are CoreSight components that contain the base addresses of all the CoreSight debug components accessible via the AHB-D. These tables allow a debugger to discover the topology of the CoreSight system automatically.

There are two ROM tables in the Cortex-M7 sub-system:

- Cortex-M7 CPU ROM table
  This table is pointed to by the base register in the Cortex-M7 AHB-AP. It contains the base address pointers for the ETM and CTI, as well as for the Cortex-M7 CPU ROM table.

- Cortex-M7 PPB (private peripheral bus) ROM table
  This table contains pointers to the Cortex-M7 system control space registers, allowing the debugger to identify the CPU core, as well as to the remaining CoreSight components in the Cortex-M7 subsystem: FPB, DWT and ITM.

The CPU ROM table occupies a 4-Kbyte, 32-bit wide chunk of AHB-D address space, from 0xE00FE000 to 0xE00FEFFC.

<table>
<thead>
<tr>
<th>Address in ROM table</th>
<th>Component name</th>
<th>Component base address</th>
<th>Component address offset</th>
<th>Size</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE00FE000</td>
<td>Cortex-M7 PPB ROM table</td>
<td>0xE00FF000</td>
<td>0x00001000</td>
<td>4 Kbyte</td>
<td>0x00001003</td>
</tr>
<tr>
<td>0xE00FE004</td>
<td>Cortex-M7 ETM</td>
<td>0xE0041000</td>
<td>0xFFF43000</td>
<td>4 Kbyte</td>
<td>0xFFF43003</td>
</tr>
<tr>
<td>0xE00FE008</td>
<td>Cortex-M7 CTI</td>
<td>0xE0043000</td>
<td>0xFFF44000</td>
<td>4 Kbyte</td>
<td>0xFFF44003</td>
</tr>
<tr>
<td>0xE00FE00C</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x1FF02002</td>
</tr>
<tr>
<td>0xE00FE010</td>
<td>Top of table</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xE00FE010 to 0xE00FEFC8</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xE00FEFCC to 0xE00FEFFC</td>
<td>ROM table registers</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>See Table 534</td>
</tr>
</tbody>
</table>

The Cortex-M7 PPB ROM table occupies a 4-Kbyte, 32-bit wide chunk of APB-D address space, from 0xE00FF000 to 0xE00FFFFC.

<table>
<thead>
<tr>
<th>Address in ROM table</th>
<th>Component name</th>
<th>Component base address</th>
<th>Component address offset</th>
<th>Size</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE00FF000</td>
<td>SCS</td>
<td>0xE000E000</td>
<td>0xFFF0F000</td>
<td>4 Kbyte</td>
<td>0xFFF0F003</td>
</tr>
<tr>
<td>0xE00FF004</td>
<td>DWT</td>
<td>0xE0001000</td>
<td>0xFFF02000</td>
<td>4 Kbyte</td>
<td>0xFFF02003</td>
</tr>
<tr>
<td>0xE00FF008</td>
<td>FPB</td>
<td>0xE0002000</td>
<td>0xFFF03000</td>
<td>4 Kbyte</td>
<td>0xFFF03003</td>
</tr>
<tr>
<td>0xE00FF00C</td>
<td>ITM</td>
<td>0xE0000000</td>
<td>0xFFF01000</td>
<td>4 Kbyte</td>
<td>0xFFF01003</td>
</tr>
<tr>
<td>0xE00FF010</td>
<td>TPIU(1)</td>
<td>0xE0040000</td>
<td>0xFFF41000</td>
<td>4 Kbyte</td>
<td>0xFFF41002</td>
</tr>
<tr>
<td>0xE00FF014</td>
<td>ETM(1)</td>
<td>0xE0041000</td>
<td>0xFFF42000</td>
<td>4 Kbyte</td>
<td>0xFFF42002</td>
</tr>
</tbody>
</table>
The topology for the CoreSight components in the Cortex-M7 subsystem is shown in **Figure 763**.

Table 533. Cortex-M7 PPB ROM table (continued)

<table>
<thead>
<tr>
<th>Address in ROM table</th>
<th>Component name</th>
<th>Component base address</th>
<th>Component address offset</th>
<th>Size</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE00FF018</td>
<td>Top of table</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xE00FF01C to 0xE00FFFFC</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xE00FFFFCC to 0xE00FFFFC</td>
<td>ROM table registers</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>See Table 535</td>
</tr>
</tbody>
</table>

1. The TPIU and ETM are included in this table by default but bit 0 is reset to indicate that they are not present.

---

### Figure 763. Cortex-M7 CoreSight Topology

[Diagram of CoreSight components and their addresses]
Cortex-M7 CPU ROM registers

**CPU ROM memory type register (M7_CPUROM_MEMTYPE)**

Address offset: 0xFCC  
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: system memory presence  
1: system memory present on this bus

**CPU ROM CoreSight peripheral identity register 4 (M7_CPUROM_PIDR4)**

Address offset: 0xFD0  
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

4KCOUNT[3:0] | JEP106CON[3:0]

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size  
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code  
0x4: Arm® JEDEC continuation code
CPU ROM CoreSight peripheral identity register 0 (M7_CPUROM_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 00C8

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number field, bits [7:0]
- 0xC8: Cortex-M7 processor ROM table

CPU ROM CoreSight peripheral identity register 1 (M7_CPUROM_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 00B4

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
- 0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number field, bits [11:8]
- 0x4: Cortex-M7 processor ROM table
**CPU ROM CoreSight peripheral identity register 2 (M7_CPUROM_PIDR2)**

Address offset: 0xFE8  
Reset value: 0x0000 000B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVISION[3:0]**: component revision number  
0x0: rev r0p0

Bit 3  **JEDEC**: JEDEC assigned value  
1: designer ID specified by JEDEC

Bits 2:0  **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]  
0x3: Arm® JEDEC code

**CPU ROM CoreSight peripheral identity register 3 (M7_CPUROM_PIDR3)**

Address offset: 0xFEC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVAND[3:0]**: metal fix version  
0x0: no metal fix

Bits 3:0  **CMOD[3:0]**: customer modified  
0x0: co customer modification
CPU ROM CoreSight component identity register 0 (M7_CPUROM_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[7:0]:** component ID field, bits [7:0]
0x0D: common ID value

CPU ROM CoreSight component identity register 1 (M7_CPUROM_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 0010

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **CLASS[3:0]:** component ID field, bits [15:12] - component class
0x1: ROM table component

Bits 3:0  **PREAMBLE[11:8]:** component ID field, bits [11:8]
0x0: common ID value
CPU ROM CoreSight component identity register 2 (M7_CPUROM_CIDR2)
Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td><strong>PREAMBLE[19:12]</strong>: component ID field, bits [23:16]</td>
</tr>
<tr>
<td></td>
<td>0x05: common ID value</td>
</tr>
</tbody>
</table>

CPU ROM CoreSight component identity register 3 (M7_CPUROM_CIDR3)
Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td><strong>PREAMBLE[27:20]</strong>: component ID field, bits [31:24]</td>
</tr>
<tr>
<td></td>
<td>0xB1: common ID value</td>
</tr>
</tbody>
</table>

Cortex-M7 CPU ROM table register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xFCC  | M7_CPUROM_MEMTYPE |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|        | Reset value    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
Table 534. Cortex-M7 CPU ROM table register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFD0</td>
<td>M7_CPUROM_PIDR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD4 to</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD0</td>
<td>M7_CPUROM_PIDR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>M7_CPUROM_PIDR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE8</td>
<td>M7_CPUROM_PIDR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE4</td>
<td>M7_CPUROM_PIDR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>M7_CPUROM_CIDR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE8</td>
<td>M7_CPUROM_CIDR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE4</td>
<td>M7_CPUROM_CIDR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>M7_CPUROM_CIDR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Cortex-M7 PPB ROM registers

**PPB ROM memory type register (M7_PPBROM_MEMTYPE)**

Address offset: 0xFCC  
Reset value: 0x0000 0001

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: system memory presence  
1: system memory present on this bus

**PPB ROM CoreSight peripheral identity register 4 (M7_PPBROM_PIDR4)**

Address offset: 0xFD0  
Reset value: 0x0000 0004

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size  
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code  
0x4: Arm® JEDEC continuation code
PPB ROM CoreSight peripheral identity register 0 (M7_PPBROM_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 00C8

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 PARTNUM[7:0]: part number field, bits [7:0]
0xC7: Cortex-M7 PPB ROM table

PPB ROM CoreSight peripheral identity register 1 (M7_PPBROM_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 00B4

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 JEP106ID[3:0]: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 PARTNUM[11:8]: part number field, bits [11:8]
0x4: Cortex-M7 PPB ROM table
### PPB ROM CoreSight peripheral identity register 2 (M7_PPROM_PIDR2)

Address offset: 0xFE8  
Reset value: 0x0000 000B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**REVISION[3:0]**  
Component revision number  
- 0x0: rev 0

**JEDEC**: JEDEC assigned value  
- 1: designer ID specified by JEDEC

- 0x3: Arm® JEDEC code

### PPB ROM CoreSight peripheral identity register 3 (M7_PPROM_PIDR3)

Address offset: 0xFEC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**REVAND[3:0]**: Metal fix version  
- 0x0: no metal fix

**CMOD[3:0]**: Customer modified  
- 0x0: no customer modifications
**PPB ROM CoreSight component identity register 0 (M7_PPBROM_CIDR0)**

Address offset: 0xFF0  
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

15 14 13 12 11 10 9  8  7  6  5  4  3  2  1  0

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[7:0]**: component ID field, bits [7:0]  
0x0D: common ID value

**PPB ROM CoreSight component identity register 1 (M7_PPBROM_CIDR1)**

Address offset: 0xFF4  
Reset value: 0x0000 0010

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

15 14 13 12 11 10 9  8  7  6  5  4  3  2  1  0

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **CLASS[3:0]**: component ID field, bits [15:12] - component class  
0x1: ROM table component

Bits 3:0  **PREAMBLE[11:8]**: component ID field, bits [11:8]  
0x0: common ID value
**PPB ROM CoreSight component identity register 2 (M7_PPBROM_CIDR2)**

Address offset: 0xFFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[19:12]**: component ID field, bits [23:16]
0x05: common ID value

**PPB ROM CoreSight component identity register 3 (M7_PPBROM_CIDR3)**

Address offset: 0xFFF0
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[27:20]**: component ID field, bits [31:24]
0xB1: common ID value

**Cortex-M7 PPB ROM table register map and reset values**

Table 535. Cortex-M7 PPB ROM table register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFCC</td>
<td>M7_PPBROM__MEMTYPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reset value
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFD0</td>
<td>M7_PPBROM_PIDR4</td>
<td>0 0 0 0 0 1 0 0</td>
<td>0xFD4</td>
<td>M7_PPBROM_PIDR0</td>
<td>1 0 1 0 0 0 1 1</td>
<td>0xFE0</td>
<td>M7_PPBROM_PIDR1</td>
<td>1 0 1 1 0 0 1 0</td>
</tr>
<tr>
<td></td>
<td>PARTNUM[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD4 to 0xFDc</td>
<td>Reserved</td>
<td></td>
<td>0xFE4</td>
<td>M7_PPBROM_PIDR2</td>
<td>0 0 0 0 0 1 0 1</td>
<td>0xFE8</td>
<td>M7_PPBROM_PIDR3</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE8</td>
<td>M7_PPBROM_PIDR2</td>
<td>0 0 0 0 0 1 0 1</td>
<td>0xFEc</td>
<td>M7_PPBROM_PIDR3</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0xFF0</td>
<td>M7_PPBROM_CIDR0</td>
<td>0 0 0 0 0 1 1 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF0</td>
<td>M7_PPBROM_CIDR0</td>
<td>0 0 0 0 0 1 1 0</td>
<td>0xFF4</td>
<td>M7_PPBROM_CIDR1</td>
<td>0 0 0 1 0 0 0 0</td>
<td>0xFF8</td>
<td>M7_PPBROM_CIDR2</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF8</td>
<td>M7_PPBROM_CIDR2</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0xFCc</td>
<td>M7_PPBROM_CIDR3</td>
<td>1 0 1 1 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
64.6.2 Cortex-M7 data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as:
- watchpoint
- ETM trigger
- PC sampling trigger
- data address sampling trigger
- data comparator (comparator 1 only)
- clock cycle counter comparator (comparator 0 only)

It also contains counters for:
- clock cycles
- folded instructions
- load store unit (LSU) operations
- sleep cycles
- number of cycles per instruction
- interrupt overhead

A DWT comparator compares one of the following with the value held in its DWT_COMP register:
- a data address
- an instruction address
- a data value
- the cycle count value, for comparator 0 only.

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:
- one or more DWT data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value
- a watchpoint debug event, on either the PC value or the accessed data address
- a CMPMATCH[N] event that signals the match outside the DWT unit

A watchpoint debug event either generates a DebugMonitor exception or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Arm®v7-M Architecture Reference Manual [5].
### Cortex-M7 DWT registers

#### DWT control register (M7_DWT_CTRL)

Address offset: 0x000  
Reset value: 0x4000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:28</td>
<td><strong>NUMCOMP[3:0]</strong>: number of comparators implemented (read-only)</td>
<td>0x4: four comparators</td>
</tr>
<tr>
<td>27</td>
<td><strong>NOTRCPKT</strong>: race sampling and exception tracing support (read-only)</td>
<td>0: supported</td>
</tr>
<tr>
<td>26</td>
<td><strong>NOEXTTRIG</strong>: external match signal, CMPMATCH support (read-only)</td>
<td>0: supported</td>
</tr>
<tr>
<td>25</td>
<td><strong>NOCYCCNT</strong>: cycle counter support (read-only)</td>
<td>0: supported</td>
</tr>
<tr>
<td>24</td>
<td><strong>NOPRFCNT</strong>: profiling counter support (read-only)</td>
<td>0: supported</td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>
| 22   | **CYCEVTENA**: enable for POSTCNT underflow event counter packet generation | 0: disabled  
|       |                                                     | 1: enabled |
| 21   | **FOLDEVTENA**: enable for folded instruction counter overflow event generation | 0: disabled  
|       |                                                     | 1: enabled |
| 20   | **LSUEVTENA**: enable for LSU counter overflow event generation | 0: disabled  
|       |                                                     | 1: enabled |
| 19   | **SLEEPEVTENA**: enable for sleep counter overflow event generation | 0: disabled  
|       |                                                     | 1: enabled |
Bit 18 **EXCEVTENA**: enable for exception overhead counter overflow event generation
   0: disabled
   1: enabled

Bit 17 **CPIEVTENA**: enable for CPI counter overflow event generation
   0: disabled
   1: enabled

Bit 16 **EXCTRCENA**: enable for exception trace generation
   0: disabled
   1: enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA**: POSTCNT counter use enable
   This bit enables the use of POSTCNT counter as a timer for periodic PC sample packet
   generation.
   0: disabled
   1: enabled

Bits 11:10 **SYNCTAP[1:0]**: position of synchronization packet counter tap on CYCCNT counter
   This field determines the synchronization packet rate.
   0x0: disabled - no synchronization packets
   0x1: tap at CYCCNT[24]
   0x2: tap at CYCCNT[26]
   0x3: tap at CYCCNT[28]

Bit 9 **CYCTAP**: position of the POSTCNT tap on the CYCCNT counter
   0: tap at CYCCNT[6]
   1: tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]**: Initial value of the POSTCNT counter
   Writes to this field are ignored if POSTCNT counter is enabled (that is, CYCEVTENA or
   PCSAMPLENA must be reset prior to writing POSTINIT).

Bits 4:1 **POSTRESET[3:0]**: reload value of the POSTCNT counter.

Bit 0 **CYCCNTENA**: CYCCNT counter enable
   0: disabled
   1: enabled

**DWT cycle count register (M7_DWT_CYCCNT)**

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| 31:0 | CYCCNT[31:0]: processor clock cycle counter |
## DWT CPI count register (M7_DWT_CPICNT)

Address offset: 0x008  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td>CPICNT[7:0]: CPI counter</td>
</tr>
<tr>
<td></td>
<td>This field counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT_LSUCNT, and counts any instruction fetch stalls.</td>
</tr>
</tbody>
</table>

### DWT exception count register (M7_DWT_EXCCNT)

Address offset: 0x00C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td>EXCCNT[7:0]: exception overhead cycle counter</td>
</tr>
<tr>
<td></td>
<td>This field counts the number of cycles spent in exception processing.</td>
</tr>
</tbody>
</table>

## DWT sleep count register (M7_DWT_SLPcnt)

Address offset: 0x010  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td>SLEEPcnt[7:0]:</td>
</tr>
<tr>
<td></td>
<td>This field counts the number of cycles spent in sleep mode.</td>
</tr>
</tbody>
</table>
DWT LSU count register (M7_DWT_LSUCNT)

Address offset: 0x014  
Reset value: 0x0000 0000

Bits 31:8: Reserved, must be kept at reset value.  
Bits 7:0: **SLEEPCNT[7:0]**: Sleep cycle counter  
This field counts the number of cycles spent in Sleep mode (WFI, WFE, sleep-on-exit).

DWT fold count register (M7_DWT_FOLDCNT)

Address offset: 0x018  
Reset value: 0x0000 0000

Bits 31:8: Reserved, must be kept at reset value.  
Bits 7:0: **LSUCNT[7:0]**: load store counter  
This field counts additional cycles required to execute load and store instructions.

Bits 31:8: Reserved, must be kept at reset value.  
Bits 7:0: **FOLDCNT[7:0]**: folded instruction counter  
This field increments on each instruction that takes 0 cycle.
### DWT program counter sample register (M7_DWT_PCSR)

Address offset: 0x01C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>EIASAMPLE[31:16]</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 EIASAMPLE[31:0]: executed instruction address sample value  
This field samples the current value of the program counter.

### DWT comparator register x (M7_DWT_COMPx)

Address offset: 0x020 + x * 0x10 (for x = 0 to 3)  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>COMP[31:16]</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 COMP[31:0]: reference value for comparison

### DWT mask register x (M7_DWT_MASKx)

Address offset: 0x024 + x * 0x10 (for x = 0 to 3)  
Reset value: 0x0000 0000

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  |  9  |  8  |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | MASK[4:0] |

| rw  | rw  | rw  | rw  | rw  |

_ST_

RM0455 Rev 3 2877/2974
DWT function register x (M7_DWT_FUNCTx)

Address offset: 0x028 + x * 0x10 (for x = 0 to 3)

Reset value: 0x0000 0000

Bits 31:5  Reserved, must be kept at reset value.

Bits 4:0  MASK[4:0]: comparator mask size

This field provides the size of the ignore mask applied to the access address for address range matching by comparator n. A debugger can write 0b11111 to this field and then read the register back to determine the maximum mask size supported.

This field provides the size of the ignore mask applied to the access address for address range matching by comparator n. A debugger can write 0b11111 to this field and then read the register back to determine the maximum mask size supported.

DWT function register x (M7_DWT_FUNCTx)

Address offset: 0x028 + x * 0x10 (for x = 0 to 3)

Reset value: 0x0000 0000

Bits 31:5  Reserved, must be kept at reset value.

Bit 24  MATCHED: comparator match (read-only)

This bit indicates if a comparator match has occurred since the register was last read.
0: no match
1: match occurred

Bits 23:20  Reserved, must be kept at reset value.

Bits 19:16  DATAVADDR1[3:0]: comparator number of a second comparator

When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a second comparator to use for linked address comparison.

Bits 15:12  DATAVADDR0[3:0]: comparator number of a comparator

When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a comparator to use for linked address comparison.

Bits 11:10  DATAVSIZE[1:0]: size of required data comparison

For data value matching, this field specifies the size of the required data comparison.
0x0: byte
0x1: half word
0x2: word
0x3: reserved

Bit 9  LNK1ENA: support of a second linked comparator (read-only)

This bit indicates whether use of a second linked comparator is supported (read-only).
1: supported
Bit 8 **DATAVMATCH**: cycle comparison enable
- 0: Performs address comparison.
- 1: Performs data value comparison.

Bit 7 **CYCMATCH**: cycle count comparison enable on comparator 0
- This bit is reserved for other comparators.
- 0: no cycle count comparison
- 1: Compares DWT_COMP0 with the cycle counter, DWT_CYCCNT.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **EMITRANGE**: data trace address offset packet enable
- This bit enables generation of data trace address offset packets (containing data address bits 0 to 15)
- 0: disabled
- 1: enabled

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 **FUNCTION[3:0]**: action on comparator match
The meaning of this field depends on the setting of the DATAVMATCH and CYCMATCH fields. See [5].

### DWT CoreSight peripheral identity register 4 (M7_DWT_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4KCOUNT[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JEP106CON[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size
- 0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
- 0x4: Arm® JEDEC code
DWT CoreSight peripheral identity register 0 (M7_DWT_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0002

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PARTNUM[7:0]**: part number field, bits [7:0]
0x02: DWT part number

DWT CoreSight peripheral identity register 1 (M7_DWT_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 00B9

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0  **PARTNUM[11:8]**: part number field, bits [11:8]
0x0: DWT part number
## DWT CoreSight peripheral identity register 2 (M7_DWT_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 003B

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Bit 23:16</th>
<th>Bit 15:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
</tr>
<tr>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
</tr>
<tr>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>1 5 1 4 1 3 1 2 1 1 1 0</td>
<td>9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVISION[3:0]**: component revision number

0x3: r0p4

Bit 3  **JEDEC**: JEDEC assigned value

1: designer ID specified by JEDEC

Bits 2:0  **JEPC106[6:4]**: JEPC106 identity code field, bits [6:4]

0x3: Arm® JEDEC code

---

## DWT CoreSight peripheral identity register 3 (M7_DWT_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Bit 23:16</th>
<th>Bit 15:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
</tr>
<tr>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
</tr>
<tr>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
</tr>
<tr>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
<td>Re.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>1 5 1 4 1 3 1 2 1 1 1 0</td>
<td>9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVAND[3:0]**: metal fix version

0x0: no metal fix

Bits 3:0  **CMOD[3:0]**: customer modified

0x0: no customer modifications
DWT CoreSight component identity register 0 (M7_DWT_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8   Reserved, must be kept at reset value.

Bits 7:0   **PREAMBLE[7:0]**: component ID field, bits [7:0]

0x0D: common ID value

DWT CoreSight component identity register 1 (M7_DWT_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 00E0

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8   Reserved, must be kept at reset value.

Bits 7:4   **CLASS[3:0]**: component ID field, bits [15:12] - component class

0xE: trace generator component

Bits 3:0   **PREAMBLE[11:8]**: component ID field, bits [11:8]

0x0: common ID value
DWT CoreSight component identity register 2 (M7_DWT_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** component ID field, bits [23:16]
0x05: common ID value

DWT CoreSight component identity register 3 (M7_DWT_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** component ID field, bits [31:24]
0xB1: common ID value
Cortex-M7 DWT register map and reset values

The Cortex-M7 DWT registers are located at address range 0xE0001000 to 0xE0001FFC, on the AHBD.

**Table 536. Cortex-M7 DWT register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>M7_DWT_CTRL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NUMCOMP[3:0]</td>
<td></td>
<td>0 1 0 0</td>
</tr>
<tr>
<td></td>
<td>NOTRCPKT</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>NOEXTRIG</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>NOCCYCNT</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>NORRFCONT</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>CYCEVTENA</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>FOLDEVTENA</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>SLEEPETENA</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>EXCEVTENA</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>EXCTRCENA</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>PCSAMPLE.NA</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>SYNCTAP[1:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>CYCTAP</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>POSTPRES[7:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>CYCTENA</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>CYCCNT[31:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x004</td>
<td>M7_DWT_CYCCNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CYCCNT[31:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>M7_DWT_CPICNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPICNT[7:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x008</td>
<td>M7_DWT_EXCCNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EXCCNT[7:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x00C</td>
<td>M7_DWT_SLPcnt</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SLEEPcnt[7:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x010</td>
<td>M7_DWT_LSUcnt</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LSUcnt[7:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x014</td>
<td>M7_DWT_FOLDcnt</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FOLDcnt[7:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x018</td>
<td>M7_DWT_PCSR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EIASAMPLE[31:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x020</td>
<td>M7_DWT_COMP0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMP[31:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x024</td>
<td>M7_DWT_MASK0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MASK[4:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x028</td>
<td>M7_DWT_FUNCt0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MATCHED</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>DATAVADDR1[3:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>DATAVADDR0[3:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>DATAVSIZE[1:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>LINKENA</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>DATAMATCH</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>CYPATCH</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>EMITRANGE</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>FUNCTION[3:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x02C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x030</td>
<td>M7_DWT_COMP1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMP[31:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x034</td>
<td>M7_DWT_MASK1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MASK[4:0]</td>
<td></td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>
### Table 536. Cortex-M7 DWT register map and reset values (continued)

| Offset   | Register name          | Description                  | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|----------|------------------------|------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x038    | M7_DWT_FUNCT1          | Reset value                  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x03C    | Reserved               |                              |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x040    | M7_DWT_COMP2           | COMP[31:0]                   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x044    | M7_DWT_MASK2           | RESET value                  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x048    | M7_DWT_FUNCT2          | MATCHED                      | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x04C    | Reserved               |                              |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x050    | M7_DWT_COMP3           | COMP[31:0]                   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x054    | M7_DWT_MASK3           | RESET value                  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x058    | M7_DWT_FUNCT3          | MATCHED                      | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x05C    | Reserved               |                              |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xFD0    | M7_DWT_PIDR4           | PARTNUM[7:0]                 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0xFD4    | Reserved               |                              |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xFE0    | M7_DWT_PIDR0           | PARTNUM[11:8]                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xFE4    | M7_DWT_PIDR1           | PARTNUM[1:0]                 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
64.6.3 Cortex-M7 instrumentation trace macrocell (ITM)

The ITM generates trace information as packets. There are four sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

1. **Software trace**
   - Software can write directly to any of 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access, and the data written, into a packet that it writes to a FIFO. The ITM outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.

2. **Hardware trace**
   - The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.

3. **Local timestamping**
   - The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.
### Cortex-M7 ITM registers

**ITM stimulus register x (M7_ITM_STIMx)**

Address offset: 0x000 + x * 0x4 (x = 0 to 31)

Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>STIMULUS[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:0 **STIMULUS[31:0]**: software event packet / FIFOREADY

Write data is output on the trace bus as a software event packet. When reading, bit 0 is a FIFOREADY indicator:

0: Stimulus port buffer is full (or port is disabled).
1: Stimulus port can accept new write data.

### ITM trace enable register (M7_ITM_TER)

Address offset: 0xE00

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>STIMENA[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>STIMENA[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:0 **STIMENA[31:0]**: stimulus port enable

Each bit n (0:31) enables the stimulus port associated with the M7_ITM_STIMn register.

0: port disabled
1: port enabled
### ITM trace privilege registers (M7_ITM_TPR)

Address offset: 0xE40  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0 PRIVMASK[3:0]: enable unprivileged access to ITM stimulus ports  
Each bit controls eight stimulus ports:  
0bXXX0: unprivileged access permitted on ports 0 to 7  
0bXXX1: only privileged access permitted on ports 0 to 7  
0bXX0X: unprivileged access permitted on ports 8 to 15  
0bXX1X: only privileged access permitted on ports 8 to 15  
0bX0XX: unprivileged access permitted on ports 16 to 23  
0bX1XX: only privileged access permitted on ports 16 to 23  
0b0XXX: unprivileged access permitted on ports 24 to 31  
0b1XXX: only privileged access permitted on ports 24 to 31

### ITM trace control register (M7_ITM_TCR)

Address offset: 0xE80  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BUSY</th>
<th>TRACEBUSID[6:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw   rw rw rw rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TSPRESCALE[1:0]</th>
<th>SWDEN</th>
<th>TXENA</th>
<th>SYNECA</th>
<th>TSENA</th>
<th>ITMENA</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:24  Reserved, must be kept at reset value.

Bit 23  **BUSY**: ITM busy
Indicates whether the ITM is currently processing events (read-only).
0: not busy
1: busy

Bits 22:16  **TRACEBUSID[6:0]**: identifier for multi-source trace stream formatting
If multi-source trace is in use, the debugger must write a non-zero value to this field.
Note: different IDs must be used for each trace source in the system.

Bits 15:10  Reserved, must be kept at reset value.

Bits 9:8  **TSPRESCALE[1:0]**: local timestamp prescale
Prescale used with the trace packet reference clock
0x0: no prescaling
0x1: Divides by 4.
0x2: Divides by 16.
0x3: Divides by 64.

Bits 7:5  Reserved, must be kept at reset value.

Bit 4  **SWOENA**: asynchronous clocking enable for the timestamp counter (read-only)
0: Timestamp counter uses processor clock.

Bit 3  **TXENA**: hardware event packet forwarding enable
This bit enables forwarding of hardware event packets from the DWT unit to the trace port.
0: disabled
1: enabled

Bit 2  **SYNCENA**: synchronization packet transmission enable
If a debugger sets this bit, it must also configure the DWT_CTRL register SYNCTAP field in
the DWT for the correct synchronization speed.
0: disabled
1: enabled

Bit 1  **TSENA**: local timestamp generation enable
0: disabled
1: enabled

Bit 0  **ITMENA**: ITM enable
0: disabled
1: enabled

### ITM CoreSight peripheral identity register 4 (M7_ITM_PIDR4)

**Address offset**: 0xFD0

**Reset value**: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4KCOUNT[3:0]</td>
<td>JEP106CON[3:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **4KCOUNT[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0  **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

**ITM CoreSight peripheral identity register 0 (M7_ITM_PIDR0)**
Address offset: 0xFE0
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PARTNUM[7:0]**: part number field, bits [7:0]
0x01: ITM part number

**ITM CoreSight peripheral identity register 1 (M7_ITM_PIDR1)**
Address offset: 0xFE4
Reset value: 0x0000 00B0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0  **PARTNUM[11:8]**: part number field, bits [11:8]
0x1: ITM part number
### ITM CoreSight peripheral identity register 2 (M7_ITM_PIDR2)

Address offset: 0xFE8  
Reset value: 0x0000 003B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number  
0x3: r0p4

Bit 3 **JEDEC**: JEDEC assigned value  
1: designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]  
0x3: Arm® JEDEC code

### ITM CoreSight peripheral identity register 3 (M7_ITM_PIDR3)

Address offset: 0xFEC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version  
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified  
0x0: no customer modifications
### ITM CoreSight component identity register 0 (M7_ITM_CIDR0)

Address offset: 0xFF0  
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

**PREAMBLE[7:0]**

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID field, bits [7:0]  
0x0D: common ID value

### ITM CoreSight component identity register 1 (M7_ITM_CIDR1)

Address offset: 0xFF4  
Reset value: 0x0000 00E0

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

**CLASS[3:0]**  
**PREAMBLE[11:8]**

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID field, bits [15:12] - component class  
0xE: trace generator component

Bits 3:0 **PREAMBLE[11:8]**: component ID field, bits [11:8]  
0x0: common ID value
ITM CoreSight component identity register 2 (M7_ITM_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 \textbf{PREAMBLE[19:12]}: component ID field, bits [23:16]
0x05: common ID value

ITM CoreSight component identity register 3 (M7_ITM_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 \textbf{PREAMBLE[27:20]}: component ID field, bits [31:24]
0xB1: common ID value

Cortex-M7 ITM register map and reset values

The ITM registers are located at address range 0xE0000000 to 0xE0000FFC, on the AHBD.

Table 537. Cortex-M7 ITM register map and reset values

| Offset        | Register name                | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------------|------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000 to 0x07C | M7_ITM_STIM0 to M7_ITM_STIM31 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|               | Reset value                  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0x080 to 0xD9C | Reserved                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xE00         | M7_ITM_TER                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|               | Reset value                  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| Offset  | Register name          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|-----------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xE40   | M7_ITM_TPR            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xE80   | M7_ITM_TCR            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xFD0   | M7_ITM_PIDR4          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value           | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFF0   | M7_ITM_PIDR0          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xFE4   | M7_ITM_PIDR1          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value           | 1  | 0  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xFE8   | M7_ITM_PIDR2          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value           | 0  | 0  | 1  | 1  | 1  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0xFEC   | M7_ITM_PIDR3          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xFF0   | M7_ITM_CIDR0          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value           | 0  | 0  | 0  | 0  | 1  | 1  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |

Table 537. Cortex-M7 ITM register map and reset values (continued)
64.6.4 Cortex-M7 breakpoint unit (FPB)

The FPB allows hardware breakpoints to be set. It contains eight comparators that monitor the instruction fetch address and return a breakpoint instruction when a match is detected. The Cortex-M7 FPB does not support flash patch functionality.

Cortex-M7 FPB registers

FPB control register (M7_FPB_CTRL)

Address offset: 0x000

Reset value: 0x0000 0080

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>bits 31:15</th>
<th>bits 14:8</th>
<th>bits 7:4</th>
<th>bits 3:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFF4</td>
<td>M7_ITM_CIDR1</td>
<td>r r r r</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF8</td>
<td>M7_ITM_CIDR2</td>
<td>r r r r</td>
<td>r r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFFC</td>
<td>M7_ITM_CIDR3</td>
<td>r r r r</td>
<td>r r</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 NUM_LIT[3:0]: number of literal address comparators supported (read-only)

0x0: no literal comparators supported.

Bits 14:12, 7:4 NUM_CODE[6:0]: instruction address comparator number field

This read-only field holds the number of instruction address comparators supported.

0x08: height instruction comparators supported

Others: reserved
Bits 3:2  Reserved, must be kept at reset value.

Bit 1  **KEY**: write protect key  
A write to M7_FPB_CTRL register is ignored if this bit is not set to 1.

Bit 0  **ENABLE**: FPB enable  
0: disabled  
1: enabled

### FPB remap register (M7_FPB_REMAP)

Address offset: 0x004  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:30  Reserved, must be kept at reset value.

Bit 29  **RMPSPT**: Flash patch remap support (read-only)  
0: remapping not supported

Bits 28:0  Reserved, must be kept at reset value.

### FPB comparator registers (M7_FPB_COMPx)

Address offset: 0x008 + x * 0x4 (for x = 0 to 7)  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPLACE[1:0]</td>
<td>Res.</td>
<td>COMP[26:14]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

COMP[13:0]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp[13:0]</td>
<td>Res.</td>
<td>ENABLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:30 REPLACE[1:0]: behavior upon COMP versus instruction fetch address match

This field defines the behavior when a match occurs between the COMP field and the instruction fetch address:
0x0: reserved
0x1: breakpoint on lower half-word, upper half-word unaffected
0x2: breakpoint on upper half-word, lower half-word unaffected
0x3: breakpoint on both upper and lower half-words

Bit 29 Reserved, must be kept at reset value.

Bits 28:2 COMP[26:0]: value to compare with code memory access address

Value to compare with address bits 28:2 of accesses to instruction code memory (0x00000000 to 0x1FFFFFFF). If a match occurs, the action to take is defined by the REPLACE field.

Bit 1 Reserved, must be kept at reset value.

Bit 0 ENABLE: comparator enable

The comparator is only enabled if both this bit and the FPB ENABLE bit in the M7_FPB_CTRL register are set.
0: disabled
1: enabled

FPB CoreSight peripheral identity register 4 (M7_FPB_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 4KCOUNT[3:0]: register file size

0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 JEP106CON[3:0]: JEP106 continuation code

0x4: Arm® JEDEC code
### FPB CoreSight peripheral identity register 0 (M7_FPB_PIDR0)

Address offset: 0xFE0  
Reset value: 0x0000 000C

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.  
Bits 7:0  **PARTNUM[7:0]**: part number field, bits [7:0]  
0x0C: FPB part number

### FPB CoreSight peripheral identity register 1 (M7_FPB_PIDR1)

Address offset: 0xFE4  
Reset value: 0x0000 00B0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.  
Bits 7:4  **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]  
0xB: Arm® JEDEC code  
Bits 3:0  **PARTNUM[11:8]**: part number field, bits [11:8]  
0x0: FPB part number
### FPB CoreSight peripheral identity register 2 (M7_FPB_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 002B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>REV</td>
<td>ITION</td>
<td>3:0</td>
<td>J</td>
<td>EDEC</td>
<td>JEP106</td>
<td>ID</td>
<td>6:4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>

 Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REV|ITION|3:0**: component revision number
0x2: r0p3

Bit 3  **JEDEC**: JEDEC assigned value
1: designer ID specified by JEDEC

Bits 2:0  **JEP106|ID|6:4**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

### FPB CoreSight peripheral identity register 3 (M7_FPB_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>REV</td>
<td>AND</td>
<td>3:0</td>
<td></td>
<td>CMOD</td>
<td>3:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>

 Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REV|AND|3:0**: metal fix version
0x0: no metal fix

Bits 3:0  **CMOD|3:0**: customer modified
0x0: no customer modifications
**FPB CoreSight component identity register 0 (M7_FPB_CIDR0)**

Address offset: 0xFF0

Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** component ID field, bits [7:0]

- 0xD: common ID value

**FPB CoreSight component identity register 1 (M7_FPB_CIDR1)**

Address offset: 0xFF4

Reset value: 0x0000 00E0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** component ID field, bits [15:12] - component class

- 0xE: trace generator component

Bits 3:0 **PREAMBLE[11:8]:** component ID field, bits [11:8]

- 0x0: common ID value
FPB CoreSight component identity register 2 (M7_FPB_CIDR2)
Address offset: 0xFF8
Reset value: 0x0000 0005

```
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  PREAMBLE[19:12]: component ID field, bits [23:16]
         0x05: common ID value
```

FPB CoreSight component identity register 3 (M7_FPB_CIDR3)
Address offset: 0xFFF
Reset value: 0x0000 00B1

```
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  PREAMBLE[27:20]: component ID field, bits [31:24]
         0xB1: common ID value
```

Cortex-M7 FPB register map and reset values
The Cortex-M7 FPB registers are located at address range 0xE0002000 to 0xE0002FFC.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>M7_FPB_CTRL</td>
<td>0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>
### Table 538. Cortex-M7 FPB register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x004</td>
<td>M7_FPB_REMAP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
</tr>
<tr>
<td>0x008 to</td>
<td>M7_FPB_COMP0 to M7_FPB_COMP7</td>
<td>0x000 to 0x020</td>
</tr>
<tr>
<td>0x020</td>
<td>COMP[26:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
</tr>
<tr>
<td>0x024 to</td>
<td>M7_FPB_PIDR4</td>
<td></td>
</tr>
<tr>
<td>0xFD0</td>
<td>M7_FPB_PIDR4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
</tr>
<tr>
<td>0xFD4 to</td>
<td>M7_FPB_PIDR0</td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>M7_FPB_PIDR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
</tr>
<tr>
<td>0xFE4</td>
<td>M7_FPB_PIDR1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>101</td>
</tr>
<tr>
<td>0xFE8</td>
<td>M7_FPB_PIDR2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>001</td>
</tr>
<tr>
<td>0xFE8</td>
<td>M7_FPB_PIDR2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>001</td>
</tr>
<tr>
<td>0xFE8</td>
<td>M7_FPB_PIDR2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>001</td>
</tr>
<tr>
<td>0xFF0</td>
<td>M7_FPB_CIDR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
</tr>
<tr>
<td>0xFF4</td>
<td>M7_FPB_CIDR1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>111</td>
</tr>
<tr>
<td>0xFF8</td>
<td>M7_FPB_CIDR2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
</tr>
<tr>
<td>0xFFC</td>
<td>M7_FPB_CIDR3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>101</td>
</tr>
</tbody>
</table>
64.6.5 Cortex-M7 embedded trace macrocell (ETM)

The Cortex-M7 ETM is a CoreSight component closely coupled to the CPU. The ETM generates trace packets that allow the execution of the Cortex-M7 core to be traced. In the STM32H7, the ETM is configured for instruction trace only, so data accesses are not included in the trace information.

The ETM receives information from the CPU over the processor trace interface, including:
- the number of instructions executed in the same cycle
- changes in program flow
- the current processor instruction state
- the addresses of memory locations accessed by load and store instructions
- the type, direction and size of a transfer
- condition code information
- exception information
- wait for interrupt state information

For more information, refer to the Arm® CoreSight ETM-M7 technical reference manual [6].

Cortex-M7 ETM registers

ETM programming control register (M7_ETM_PRGCTL)

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: trace program enable
0: trace unit disabled
1: trace unit enabled
**ETM status register (M7_ETM_STAT)**

Address offset: 0x00C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-2</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

- **Bit 1 PMSTABLE**: programmers model stable
  - This bit indicates whether the ETM registers are stable and can be read.
  - 0: registers not stable
  - 1: registers stable

- **Bit 0 IDLE**: trace unit inactive
  - 0: ETM not idle
  - 1: ETM idle

**ETM trace configuration register (M7_ETM_CONFIG)**

Address offset: 0x010

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-18</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

- **Bit 17 DV**: data value tracing (read-only)
  - 0: disabled

- **Bit 16 DA**: data address tracing (read-only)
  - 0: disabled

- **Bits 15-13** Reserved, must be kept at reset value.

- **Bit 12 RS**: return stack enable
  - 0: disabled
  - 1: enabled
Bit 11 **TS**: global timestamp tracing  
0: disabled  
1: enabled  
Note: global timestamp not implemented in this device

Bits 10:8 **COND[2:0]**: conditional instruction tracing  
0x0: conditional instruction tracing disabled  
0x1: conditional load instructions traced  
0x2: conditional store instructions traced  
0x3: conditional load and store instructions traced  
0x7: all conditional instructions traced  
Other: reserved

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CCI**: cycle counting in instruction trace  
0: disabled  
1: enabled

Bit 3 **BB**: branch broadcast mode  
0: disabled  
1: enabled

Bits 2:1 **INSTP0[1:0]**: determines which instructions are P0 instructions (read-only)  
0x0: Only branches are P0 instructions.

Bit 0 Reserved, must be kept at reset value.

**ETM event control 0 register (M7_ETM_EVENTCTL0)**

Only accepts writes when trace unit is disabled  
Address offset: 0x020  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TYPE1</th>
<th>SEL1[3:0]</th>
<th>TYPE0</th>
<th>SEL0[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TYPE1**: resource type for event 1  
0: single selected resource  
1: boolean combined resource pair

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:8 **SEL[3:0]**: resource/boolean combined resource pair, for event 1  
When **TYPE1** is 0, this field selects a single selected resource from 0-15 defined by bits[3:0].  
When **TYPE1** is 1, this field selects a boolean combined resource pair from 0-7 defined by bits[2:0].
Bit 7 **TYPE0**: resource type for event 0
   0: single selected resource
   1: boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL0[3:0]**: resource/boolean combined resource pair for event 0
   When TYPE0 is 0, this field selects a single selected resource from 0-15 defined by bits[3:0].
   When TYPE0 is 1, this field selects a boolean combined resource pair from 0-7 defined by bits[2:0].

**ETM event control 1 register (M7_ETM_EVENTCTL1)**

Only accepts writes when trace unit is disabled

Address offset: 0x024

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LPOVERRIDE**: low-power state behavior override
   0: low-power state normal behavior
   1: Entry to low power state does not affect resources and event trace generation.

Bit 11 **ATB**: ATB trigger enable
   0: disabled
   1: enabled

Bits 10:4 Reserved, must be kept at reset value.

Bits 3:0 **INSTEN[3:0]**: instruction trace event element enable
   Each bit corresponds to an event:
   0bXXX0: Event 0 does not cause an event element.
   0bXXX1: Event 0 causes an event element.
   0bXX0X: Event 1 does not cause an event element.
   0bXX1X: Event 1 causes an event element.
   0bX0XX: Event 2 does not cause an event element.
   0bX1XX: Event 2 causes an event element.
   0b0XXX: Event 3 does not cause an event element.
   0b1XXX: Event 3 causes an event element.
ETM stall control register (M7_ETM_STALLCTL)

Only accepts writes when trace unit is disabled

Address offset: 0x02C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

| rw | rw | rw |

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 DSTALL: stalls processor based on data trace buffer space
0: Do not stall processor.
1: Stalls processor.

Bit 8 ISTALL: Stalls processor based on instruction trace buffer space
0: Do not stall processor.
1: Stalls processor.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 LEVEL[1:0]: Stalling threshold level
A low level minimizes the amount of processor stalling, with a higher risk of FIFO overflow. A high level minimizes the risk of FIFO overflow but increases the amount of processor stalling.

Bits 1:0 Reserved, must be kept at reset value.

ETM synchronization period register (M7_ETM_SYNCP)

Address offset: 0x034
Reset value: 0x0000 000A

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

| r | r | r | r | r |

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 PERIOD[4:0]: trace bytes between synchronization requests
This field defines the number of bytes of trace information between trace synchronization requests.
0xA: 1024 bytes
ETM cycle count control register (M7_ETM_CCCTL)

Address offset: 0x038
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:12  Reserved, must be kept at reset value.

Bits 11:0  THRESHOLD[11:0]: threshold value for instruction trace cycle counting
The threshold represents the minimum interval between cycle count trace packets.
0x0: reserved
Others: threshold

ETM trace ID register (M7_ETM_TRACEID)

Address offset: 0x040
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:7  Reserved, must be kept at reset value.

Bits 6:0  TRACEID[6:0]: trace ID
0x00: reserved
0x01 to 0x6F: valid ID
0x70 to 0x7F: reserved
ETM ViewInst main control register (M7_ETM_VICTL)

Address offset: 0x080
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit(s)</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-20</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>EXLEVEL_S3: trace disable, exception level 3</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit disables tracing in the specified</td>
<td></td>
</tr>
<tr>
<td></td>
<td>exception level in Secure state for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>exception level 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enables ViewInst in this exception level.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disables ViewInst in this exception level.</td>
<td></td>
</tr>
<tr>
<td>18-17</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>EXLEVEL_S0: trace disable, exception level 0</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit disables tracing in the specified</td>
<td></td>
</tr>
<tr>
<td></td>
<td>exception level in Secure state for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>exception level 0.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enables ViewInst in this exception level.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disables ViewInst in this exception level.</td>
<td></td>
</tr>
<tr>
<td>15-12</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>TRCERR: tracing of system error exception</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit selects whether a system error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>exception must always be traced.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: System error exception is traced only if</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the instruction or exception immediately before</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the system error exception is traced.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: System error exception is always traced</td>
<td></td>
</tr>
<tr>
<td></td>
<td>regardless of the value of ViewInst.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>TRCRESET: tracing of reset exception</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit selects whether a reset exception</td>
<td></td>
</tr>
<tr>
<td></td>
<td>must always be traced.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Reset exception is traced only if the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>instruction or exception immediately before the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset exception is traced.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Reset exception is always traced</td>
<td></td>
</tr>
<tr>
<td></td>
<td>regardless of the value of ViewInst.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>SSSTATUS: current status of the start/stop logic</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>0: Stop state</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Started state</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
</tbody>
</table>
### ETM ViewInst start/stop control register (M7_ETM_VISSCTL)

Address offset: 0x088

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Stop[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24</td>
<td>rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8</td>
<td>rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 7:0</th>
<th>Start[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td>rw rw rw rw</td>
</tr>
</tbody>
</table>

### ETM ViewInst start/stop processor comparator control register (M7_ETM_VIPCSSCTL)

Address offset: 0x08C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Stop[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24</td>
<td>rw rw rw rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8</td>
<td>rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 7:0</th>
<th>Start[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td>rw rw rw rw</td>
</tr>
</tbody>
</table>

---

Bit 7 **TYPE**: resource type
- 0: single selected resource
- 1: boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]**: resource/boolean combined resource pair
- When TYPE is 0, this field selects a single selected resource from 0-15 defined by bits[3:0].
- When TYPE is 1, this field selects a boolean combined resource pair from 0-7 defined by bits[2:0].
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:16  STOP[3:0]: selector of processor comparator input to stop trace
This field selects which processor comparator inputs are in use with ViewInst Start/Stop control, for the purpose of stopping trace. One bit is provided for each processor comparator input.

Bits 15:4  Reserved, must be kept at reset value.

Bits 3:0  START[3:0]: selector of processor comparator input to start trace
This field selects which processor comparator inputs are in use with ViewInst Start/Stop control, for the purpose of starting trace. One bit is provided for each processor comparator input.

ETM counter reload value register (M7_ETM_CNTRLDV)

Address offset: 0x140
Reset value: 0x0000 0000

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  VALUE[15:0]: counter reload value
This value is loaded into the counter each time the reload event occurs.

ETM ID register 8 (M7_ETM_IDR8)

Address offset: 0x180
Reset value: 0x0000 0002

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  VALUE[15:0]: counter reload value
This value is loaded into the counter each time the reload event occurs.

Bits 31:0  MAXSPEC[31:0]: maximum speculation depth
This field indicates the maximum speculation depth of the instruction trace stream. This is the maximum number of P0 elements that have not been committed in the trace stream at any one time.
0x2: maximum trace speculation depth is 2
ETM ID register 9 (M7_ETM_IDR9)
Address offset: 0x184
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:0 NUMP0KEY[31:0]: number of P0 right-hand keys used**
- 0x0: no P0 keys used in instruction trace only configuration

ETM ID register 10 (M7_ETM_IDR10)
Address offset: 0x188
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:0 NUMP1KEY[31:0]: total number of P1 right-hand keys**
- This field indicates the total number of P1 right-hand keys, including normal and special keys.
- 0x0: No P1 keys used in instruction trace only configuration

ETM ID register 11 (M7_ETM_IDR11)
Address offset: 0x18C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:0 NUMP1SPC[31:0]: total number of special P1 right-hand keys used**
- 0x0: no special P1 keys used
**ETM ID register 12 (M7_ETM_IDR12)**

Address offset: 0x190  
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NUMCONDKEY[31:16]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**Bits 31:0 NUMCONDKEY[31:0]:**

This field indicates the total number of conditional instruction right-hand keys, including normal and special keys.  
0x1: one conditional instruction right hand-key implemented

**ETM ID register 13 (M7_ETM_IDR13)**

Address offset: 0x194  
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NUMCONDSPC[31:16]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**Bits 31:0 NUMCONDSPC[31:0]:** number of special conditional instruction right-hand keys  
0x0: no special conditional instruction right hand-keys implemented

**ETM implementation specific register 0 (M7_ETM_IMSPEC0)**

Address offset: 0x1C0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SUPPORT[3:0]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**RM0455 Debug infrastructure**
ETM ID register 0 (M7_ETM_IDR0)

Address offset: 0x1E0
Reset value: 0x0C00 1EE1

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  SUPPORT[3:0]: support for implementation specific extensions
0x0: no implementation specific extensions are supported

ETM ID register 0 (M7_ETM_IDR0)

Address offset: 0x1E0
Reset value: 0x0C00 1EE1

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:30  Reserved, must be kept at reset value.

Bit 29  COMMOPT: meaning of the commit field in some packets
0: commit mode 0

Bits 28:24  TSIZE[4:0]: global timestamp size
0x08: maximum of 64-bit global timestamp implemented.
Note: global timestamp not implemented in this device

Bits 23:17  Reserved, must be kept at reset value.

Bits 16:15  QSUPP[1:0]: Q element support
0x0: Q elements not supported

Bit 14  Reserved, must be kept at reset value.

Bits 13:12  CONDTYPE[1:0]: way of conditional result tracing
0x1: APSR condition flag values traced

Bits 11:10  NUMEVENT[1:0]: number of events supported in the trace
0x1: two events supported for instruction only configuration

Bit 9  RETSTACK: return stack support
1: two entry return stack supported

Bit 8  Reserved, must be kept at reset value.

Bit 7  TRCCCI: support for cycle counting in the instruction trace
1: cycle counting in the instruction trace implemented
Bit 6 **TRCCOND**: support for conditional instruction tracing  
1: conditional instruction trace implemented

Bit 5 **TRCBB**: support for branch broadcast tracing  
1: branch broadcast trace implemented

Bits 4:0 Reserved, must be kept at reset value.

**ETM ID register 1 (M7_ETM_IDR1)**
Address offset: 0x1E4
Reset value: 0x4100 F401

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 **DESIGNER[7:0]**: trace unit designer entity  
0x41: Arm®

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:8 **TRCARCHMAJ[3:0]**: major trace unit architecture version number  
0x4: ETM v4

Bits 7:4 **TRCARCHMIN[3:0]**: minor trace unit architecture version number  
0x0: minor version 0

Bits 3:0 **REVISION[3:0]**: implementation revision number  
0x1: rev 1

**ETM ID register 2 (M7_ETM_IDR2)**
Address offset: 0x1E8
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMIDSIZE[4:0]</td>
<td>CIDSIZE[4:0]</td>
<td>IASIZE[4:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
</tbody>
</table>
ETM ID register 3 (M7_ETM_IDR3)

Address offset: 0x1EC
Reset value: 0x0509 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31 **NOOVERFLOW**: support of NOOVERFLOW
This bit indicates whether the NOOVERFLOW of trace stall control is implemented. 0: not implemented

Bits 30:28 **NUMPROC[2:0]**: number of processors available for tracing 0x0: Only one processor can be traced.

Bit 27 **SYSTALL**: system support for stall control of the processor 0: not supported

Bit 26 **STALLCTL**: stall control support 1: trace stall control (TRCSTALLCTL) implemented

Bit 25 **SYNCPRI**: trace synchronization period support 0: TRCSYNCPRI is read-only for instruction trace only configuration; the trace synchronization period is fixed.
Bit 24  **TRCERR**: support of TRCVICTLR.TRCERR
    This bit indicates whether TRCVICTLR.TRCERR is implemented.
    0x4: 32-bit maximum address size

Bits 23:20  Reserved, must be kept at reset value.

Bits 19:16  **EXLEVEL_S[3:0]**: support of privilege levels
    Privilege levels are implemented; one bit for each level.
    0x9: privilege levels thread and handler implemented

Bits 15:12  Reserved, must be kept at reset value.

Bits 11:0  **CCITMIN[11:0]**: instruction trace cycle counting minimum threshold
    0x4: Minimum threshold is 4 instruction trace cycle.

**ETM ID register 4 (M7_ETM>IDR4)**

Address offset: 0x1F0

Reset value: 0x0001 4000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

Bits 31:28  **NUMVMIDC[3:0]**: number of virtual machine ID comparators implemented
    0x0: none

Bits 27:24  **NUMCIDC[3:0]**: number of context ID comparators implemented
    0x0: none

Bits 23:20  **NUMSSCC[3:0]**: number of single-shot comparator controls implemented
    0x0: none

Bits 19:16  **NUMRSPAIR[3:0]**: number of resource selection pairs implemented
    0x1: none

Bits 15:12  **NUMPC[3:0]**: number of processor comparator inputs implemented
    0x4: four

Bits 11:9  Reserved, must be kept at reset value.

Bit 8  **SUPPADC**: support of data address comparisons
    0: not implemented

Bits 7:4  **NUMDVC[3:0]**: number of data value comparators implemented
    0x0: none

Bits 3:0  **NUMACPAIRS[3:0]**: number of address comparator pairs implemented.
    0x0: none
**ETM ID register 5 (M7_ETM_IDR5)**

Address offset: 0x1F4

Reset value: 0x90C7 0402

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>REDFUNCNTR: support of reduced function counter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1: implemented</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 30:28</th>
<th>NUMCNTR[2:0]: number of counters implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1:</td>
<td>one counter implemented</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 27:25</th>
<th>NUMSEQSTATE[2:0]: number of sequencer states implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0:</td>
<td>none</td>
</tr>
</tbody>
</table>

| Bit 24 | Reserved, must be kept at reset value.                  |

<table>
<thead>
<tr>
<th>Bit 23</th>
<th>LPOVERRIDE: support of low-power state override</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>implemented</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 22</th>
<th>ATBTRIG: support of ATB trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>implemented</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 21:16</th>
<th>TRACEIDSIZE[5:0]: number of bits of trace ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x07:</td>
<td>seven-bit trace ID implemented.</td>
</tr>
</tbody>
</table>

| Bits 15:12 | Reserved, must be kept at reset value.       |

<table>
<thead>
<tr>
<th>Bits 11:9</th>
<th>NUMEXTINSEL[2:0]: number of external input selectors implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2:</td>
<td>two external input selectors implemented</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 8:0</th>
<th>NUMEXTIN[8:0]: number of external inputs implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2:</td>
<td>two external inputs implemented</td>
</tr>
</tbody>
</table>
**ETM resource selection register 2 (M7_ETM_RSCTL2)**

Address offset: 0x208

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:22  Reserved, must be kept at reset value.

- **Bit 21** **PAIRINV**: inversion of result of a combined pair of resources
  - 0: not inverted
  - 1: inverted

- **Bit 20** **INV**: inversion of the selected resources
  - 0: not inverted
  - 1: inverted

- **Bit 19** Reserved, must be kept at reset value.

- **Bits 18:16** **GROUP[2:0]**: selects a group of resources
  - Bits 15:8 Reserved, must be kept at reset value.

- **Bits 7:0** **SELECT[7:0]**: selector of resources from desired group
  
  This field selects one or more resources from the desired group. One bit is provided per resource from the group.

**ETM resource selection register 3 (M7_ETM_RSCTL3)**

Address offset: 0x20C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:21  Reserved, must be kept at reset value.

- **Bit 20** **INV**: inversion of the selected resources
  - 0: not inverted
  - 1: inverted

- **Bit 19** Reserved, must be kept at reset value.
Bits 18:16  **GROUP[2:0]**: selects a group of resources

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:0  **SELECT[7:0]**: selector of resources from desired group

This field selects one or more resources from the desired group. One bit is provided per resource from the group.

**ETM single-shot comparator control register 0 (M7_ETM_SSCC0)**

Address offset: 0x280

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ETM single-shot comparator status register 0 (M7_ETM_SSCS0)**

Address offset: 0x2A0

Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 24  **RST**: single-shot comparator resource reset enable

This bit enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected.

0: disabled
1: reset enabled; multiple matches can occur.

Bits 23:0  Reserved, must be kept at reset value.
Bit 31 **STATUS**: single-shot status

This bit indicates whether any of the selected comparators have matched. If SSCC0.RST is set to 0, the STATUS bit must be written with 0 in order to enable single-shot comparator control.

- 0: no match occurred
- 1: match occurred at least once

Bits 30:3 Reserved, must be kept at reset value.

Bit 2 **DV**: data value comparator support

0: single-shot data value comparisons not supported

Bit 1 **DA**: data address comparator support

0: single-shot data address comparisons not supported

Bit 0 **INST**: instruction address comparator support

1: single-shot instruction address comparisons supported

**ETM single-shot processor comparator input control register**

(M7_ETM_SSPCIC0)

Address offset: 0x2C0

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>STATUS: single-shot status</td>
</tr>
<tr>
<td>30:3</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>2</td>
<td>DV: data value comparator support</td>
</tr>
<tr>
<td>1</td>
<td>DA: data address comparator support</td>
</tr>
<tr>
<td>0</td>
<td>INST: instruction address comparator support</td>
</tr>
</tbody>
</table>

**PC[7:0]**: comparator input selector for single-shot control

This field selects one or more processor comparator inputs for single-shot control. One bit is provided for each processor comparator input.

**ETM power-down control register** (M7_ETM_PDC)

Address offset: 0x310

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30:0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
ETM power-down status register (M7_ETM_PDS)

Address offset: 0x314
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>Bit 31-24</th>
<th>Bit 23-16</th>
<th>Bit 15-8</th>
<th>Bit 7-4</th>
<th>Bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Bit 3 PU**: power up request
- Request to maintain power to the ETM and access to the trace registers.
- 0: power not requested
- 1: power requested

**Bits 2:0**: Reserved, must be kept at reset value.

ETM claim tag set register (M7_ETMCLAIMSET)

Address offset: 0xFA0
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>Bit 31-24</th>
<th>Bit 23-16</th>
<th>Bit 15-8</th>
<th>Bit 7-4</th>
<th>Bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Bit 1 STICKYPD**: Sticky power-down state
- This bit is set to 1 when power to the ETM registers is removed, to indicate that programming state has been lost. It is cleared after a read of the TRCPDSR.
- 0: trace register power uninterrupted since the last read of PDS register
- 1: trace register power interrupted since the last read of PDS register

**Bit 0 POWER**: ETM powered up
- 1: ETM is powered up; all registers are accessible.
**ETM claim tag clear register (M7_ETM_CLAIMCLR)**

Address offset: 0xFA4
Reset value: 0x0000 0000

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **CLAIMCLR[3:0]**: Resets claim tag bits

Write:
- 0000: no effect
- xxx1: Clears bit 0.
- xx1x: Clears bit 1.
- x1xx: Clears bit 2.
- 1xxx: Clears bit 3.

Read: Returns current value of claim tag.

---

**ETM lock access register (M7_ETM_LAR)**

Address offset: 0xFB0
Reset value: 0xFFFF XXXX

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **CLAIMCLR[3:0]**: Resets claim tag bits

Write:
- 0000: no effect
- xxx1: Clears bit 0.
- xx1x: Clears bit 1.
- x1xx: Clears bit 2.
- 1xxx: Clears bit 3.

Read: Returns current value of claim tag.
ETM lock status register (M7_ETM_LSR)
Address offset: 0xFB4
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **ACCESS_W[31:0]**: ETM register write access
- This field enables write access to some ETM registers by processor cores (debuggers do not need to unlock the component)
- 0xC5ACCE55: write access enabled
- Other values: write access disabled

**ETM encryption status register (M7_ETM_AUTHSTAT)**
Address offset: 0xFB8
Reset value: 0x0000 000A

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:3: Reserved, must be kept at reset value.

- Bit 2 **LOCKTYPE**: size of the M7_ETM_LAR register
  - 0: 32-bit
- Bit 1 **LOCKGRANT**: current status of lock
  - This bit always returns zero when read by an external debugger.
  - 0: Write access permitted
  - 1: Write access blocked. Only read access is permitted.

- Bit 0 **LOCKEXIST**: existence of lock control mechanism
  - The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.
  - 0: no lock control mechanism
  - 1: lock control mechanism implemented

2924/2974 RM0455 Rev 3
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:6  **SNID[1:0]**: security level for secure non-invasive debug  
0x0: not implemented

Bits 5:4  **SID[1:0]**: security level for secure invasive debug  
0x0: not implemented

Bits 3:2  **NSNID[1:0]**: security level for non-secure non-invasive debug  
0x2: disabled  
0x3: enabled

Bits 1:0  **NSID[1:0]**: security level for non-secure invasive debug  
0x2: disabled  
0x3: enabled

**ETM CoreSight device architecture register (M7_ETM_DEVARCH)**

Address offset: 0xFBC  
Reset value: 0x4770 4A13

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**ARCHITECT[10:0]**: component architect  
0x23B: Arm®

Bit 20  **PRESENT**: indicates the presence of this register  
1: present

Bits 19:16  **REVISION[3:0]**: architecture revision  
0x0: rev 0

Bits 15:0  **ARCHID[15:0]**: architecture ID  
0x4A13: ETMv4 component
**ETM CoreSight device type identity register (M7_ETM_DEVTYPE)**

Address offset: 0xFCC
Reset value: 0x0000 0013

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>SUBTYPE[3:0]</th>
<th>MAJORTYPE[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:8</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>
| Bits 7:4    | **SUBTYPE[3:0]**: device sub-type identifier  
0x1: processor trace |
| Bits 3:0    | **MAJORTYPE[3:0]**: device main type identifier  
0x3: trace source |

**ETM CoreSight peripheral identity register 4 (M7_ETM_PIDR4)**

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>4KCOUNT[3:0]</th>
<th>JEP106CON[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:8</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>
| Bits 7:4    | **4KCOUNT[3:0]**: register file size  
0x0: The register file occupies a single 4-Kbyte region. |
| Bits 3:0    | **JEP106CON[3:0]**: JEP106 continuation code  
0x4: Arm® JEDEC code |
ETM CoreSight peripheral identity register 0 (M7_ETM_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0075

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td><strong>PARTNUM[7:0]</strong>: part number field, field, bits [7:0]</td>
</tr>
<tr>
<td></td>
<td>0x75: ETM part number</td>
</tr>
</tbody>
</table>

ETM CoreSight peripheral identity register 1 (M7_ETM_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 00B9

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:4</td>
<td><strong>JEP106ID[3:0]</strong>: JEP106 identity code field, bits [3:0]</td>
</tr>
<tr>
<td></td>
<td>0xB: Arm® JEDEC code</td>
</tr>
<tr>
<td>Bits 3:0</td>
<td><strong>PARTNUM[11:8]</strong>: part number field, bits [11:8]</td>
</tr>
<tr>
<td></td>
<td>0x9: ETM part number</td>
</tr>
</tbody>
</table>
ETM CoreSight peripheral identity register 2 (M7_ETM_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 002B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVISION[3:0]**: component revision number
0x1: r0p2

Bit 3  **JEDEC**: JEDEC assigned value
1: designer ID specified by JEDEC

Bits 2:0  **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

ETM CoreSight peripheral identity register 3 (M7_ETM_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0  **CMOD[3:0]**: customer modified
0x0: no customer modifications
ETM CoreSight component identity register 0 (M7_ETM_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  PREAMBLE[7:0]: component ID field, bits [7:0]
0x0D: common ID value

ETM CoreSight component identity register 1 (M7_ETM_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 0090

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  CLASS[3:0]: component ID field, bits [15:12] - component class
0x09: debug component with CoreSight-compatible registers

Bits 3:0  PREAMBLE[11:8]: component ID field, bits [11:8]
0x0: common ID value
ETM CoreSight component identity register 2 (M7_ETM_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[19:12]**: component ID field, bits [23:16]
0x05: common ID value

ETM CoreSight component identity register 3 (M7_ETM_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[27:20]**: component ID field, bits [31:24]
0xB1: common ID value

Cortex-M7 ETM register map and reset values

The ETM registers are accessed by the debugger via the Cortex-M7 PPB, at address range 0xE0041000 to 0xE0041FFC.

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x004  | M7_ETM_PRGCTL | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  | R  |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Table 539. Cortex-M7 ETM register map and reset values
### Table 539. Cortex-M7 ETM register map and reset values (continued)

<p>| Offset | Register name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x00C  | M7_ETM_STAT   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x010  | M7_ETM_CONFIG |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x014 to 0x01C | Reserved     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x020  | M7_ETM_EVTNCTL0 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x024  | M7_ETM_EVTNCTL1 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x02C  | M7_ETM_STALLCTL |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x030  | Reserved      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x034  | M7_ETM_SYNCP  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x038  | M7_ETM_CCCTL  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x040  | M7_ETM_TRACEID |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x080  | M7_ETM_VICTL  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x088  | M7_ETM_VISSCTL |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x08C  | M7_ETM_VICPSSCTL |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |</p>
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset Register name</th>
<th>Offset Register name</th>
<th>Offset Register name</th>
<th>Offset Register name</th>
<th>Offset Register name</th>
<th>Offset Register name</th>
<th>Offset Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x140</td>
<td>M7_ETM_CNTRLDV</td>
<td>VALUE[15:0]</td>
<td>Reset value</td>
<td>0x140</td>
<td>M7_ETM_CNTRLDV</td>
<td>VALUE[15:0]</td>
<td>Reset value</td>
<td>0x140</td>
</tr>
<tr>
<td>0x144</td>
<td>M7_ETM_IIDR8</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x144</td>
<td>M7_ETM_IIDR8</td>
<td>MAXSPEC[31:0]</td>
<td>Reset value</td>
<td>0x144</td>
</tr>
<tr>
<td>0x148</td>
<td>M7_ETM_IIDR9</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x148</td>
<td>M7_ETM_IIDR9</td>
<td>NUMP0KEY[31:0]</td>
<td>Reset value</td>
<td>0x148</td>
</tr>
<tr>
<td>0x152</td>
<td>M7_ETM_IIDR10</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x152</td>
<td>M7_ETM_IIDR10</td>
<td>NUMP1KEY[31:0]</td>
<td>Reset value</td>
<td>0x152</td>
</tr>
<tr>
<td>0x156</td>
<td>M7_ETM_IIDR11</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x156</td>
<td>M7_ETM_IIDR11</td>
<td>NUMP1SPC[31:0]</td>
<td>Reset value</td>
<td>0x156</td>
</tr>
<tr>
<td>0x160</td>
<td>M7_ETM_IIDR12</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x160</td>
<td>M7_ETM_IIDR12</td>
<td>NUMCONDKEY[31:0]</td>
<td>Reset value</td>
<td>0x160</td>
</tr>
<tr>
<td>0x164</td>
<td>M7_ETM_IIDR13</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x164</td>
<td>M7_ETM_IIDR13</td>
<td>NUMCONDSPC[31:0]</td>
<td>Reset value</td>
<td>0x164</td>
</tr>
<tr>
<td>0x168</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x168</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x168</td>
</tr>
<tr>
<td>0x172</td>
<td>M7_ETM_IIMSPEC0</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x172</td>
<td>M7_ETM_IIMSPEC0</td>
<td>SUPPORT[3:0]</td>
<td>Reset value</td>
<td>0x172</td>
</tr>
<tr>
<td>0x176</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x176</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x176</td>
</tr>
<tr>
<td>0x180</td>
<td>M7_ETM_IIDR8</td>
<td>MAXSPEC[31:0]</td>
<td>Reset value</td>
<td>0x180</td>
<td>M7_ETM_IIDR8</td>
<td>MAXSPEC[31:0]</td>
<td>Reset value</td>
<td>0x180</td>
</tr>
<tr>
<td>0x184</td>
<td>M7_ETM_IIDR9</td>
<td>NUMP0KEY[31:0]</td>
<td>Reset value</td>
<td>0x184</td>
<td>M7_ETM_IIDR9</td>
<td>NUMP0KEY[31:0]</td>
<td>Reset value</td>
<td>0x184</td>
</tr>
<tr>
<td>0x188</td>
<td>M7_ETM_IIDR10</td>
<td>NUMP1KEY[31:0]</td>
<td>Reset value</td>
<td>0x188</td>
<td>M7_ETM_IIDR10</td>
<td>NUMP1KEY[31:0]</td>
<td>Reset value</td>
<td>0x188</td>
</tr>
<tr>
<td>0x192</td>
<td>M7_ETM_IIDR11</td>
<td>NUMP1SPC[31:0]</td>
<td>Reset value</td>
<td>0x192</td>
<td>M7_ETM_IIDR11</td>
<td>NUMP1SPC[31:0]</td>
<td>Reset value</td>
<td>0x192</td>
</tr>
<tr>
<td>0x196</td>
<td>M7_ETM_IIDR12</td>
<td>NUMCONDKEY[31:0]</td>
<td>Reset value</td>
<td>0x196</td>
<td>M7_ETM_IIDR12</td>
<td>NUMCONDKEY[31:0]</td>
<td>Reset value</td>
<td>0x196</td>
</tr>
<tr>
<td>0x1A0</td>
<td>M7_ETM_IIDR13</td>
<td>NUMCONDSPC[31:0]</td>
<td>Reset value</td>
<td>0x1A0</td>
<td>M7_ETM_IIDR13</td>
<td>NUMCONDSPC[31:0]</td>
<td>Reset value</td>
<td>0x1A0</td>
</tr>
<tr>
<td>0x1A4</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x1A4</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x1A4</td>
</tr>
<tr>
<td>0x1A8</td>
<td>M7_ETM_IIDR0</td>
<td>COMMOPT</td>
<td>TSIZE[4:0]</td>
<td>0x1A8</td>
<td>M7_ETM_IIDR0</td>
<td>COMMOPT</td>
<td>TSIZE[4:0]</td>
<td>0x1A8</td>
</tr>
<tr>
<td>0x1B2</td>
<td>M7_ETM_IIDR1</td>
<td>DESIGNER[7:0]</td>
<td>QSPAP[1:0]</td>
<td>0x1B2</td>
<td>M7_ETM_IIDR1</td>
<td>DESIGNER[7:0]</td>
<td>QSPAP[1:0]</td>
<td>0x1B2</td>
</tr>
<tr>
<td>0x1B6</td>
<td>M7_ETM_IIDR2</td>
<td>CCSIZE[3:0]</td>
<td>DVSIZE[4:0]</td>
<td>0x1B6</td>
<td>M7_ETM_IIDR2</td>
<td>CCSIZE[3:0]</td>
<td>DVSIZE[4:0]</td>
<td>0x1B6</td>
</tr>
<tr>
<td>0x1B0</td>
<td>M7_ETM_IIDR3</td>
<td>NOOPEFLOW</td>
<td>NUMPROCOC12[3:0]</td>
<td>0x1B0</td>
<td>M7_ETM_IIDR3</td>
<td>NOOPEFLOW</td>
<td>NUMPROCOC12[3:0]</td>
<td>0x1B0</td>
</tr>
<tr>
<td>0x1B4</td>
<td>M7_ETM_IIDR1</td>
<td>TRCRARCHMAJ[3:0]</td>
<td>TRCRARCHMIN[3:0]</td>
<td>0x1B4</td>
<td>M7_ETM_IIDR1</td>
<td>TRCRARCHMAJ[3:0]</td>
<td>TRCRARCHMIN[3:0]</td>
<td>0x1B4</td>
</tr>
<tr>
<td>0x1C0</td>
<td>M7_ETM_IIMSPEC0</td>
<td>SUPPORT[3:0]</td>
<td>REFISON[15:0]</td>
<td>0x1C0</td>
<td>M7_ETM_IIMSPEC0</td>
<td>SUPPORT[3:0]</td>
<td>REFISON[15:0]</td>
<td>0x1C0</td>
</tr>
<tr>
<td>0x1C4</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x1C4</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x1C4</td>
</tr>
<tr>
<td>0x1C8</td>
<td>M7_ETM_IIDR0</td>
<td>COMMOPT</td>
<td>TSIZE[4:0]</td>
<td>0x1C8</td>
<td>M7_ETM_IIDR0</td>
<td>COMMOPT</td>
<td>TSIZE[4:0]</td>
<td>0x1C8</td>
</tr>
<tr>
<td>0x1CC</td>
<td>M7_ETM_IIDR1</td>
<td>DESIGNER[7:0]</td>
<td>QSPAP[1:0]</td>
<td>0x1CC</td>
<td>M7_ETM_IIDR1</td>
<td>DESIGNER[7:0]</td>
<td>QSPAP[1:0]</td>
<td>0x1CC</td>
</tr>
<tr>
<td>0x1D0</td>
<td>M7_ETM_IIDR2</td>
<td>CCSIZE[3:0]</td>
<td>DVSIZE[4:0]</td>
<td>0x1D0</td>
<td>M7_ETM_IIDR2</td>
<td>CCSIZE[3:0]</td>
<td>DVSIZE[4:0]</td>
<td>0x1D0</td>
</tr>
<tr>
<td>0x1D4</td>
<td>M7_ETM_IIDR3</td>
<td>NOOPEFLOW</td>
<td>NUMPROCOC12[3:0]</td>
<td>0x1D4</td>
<td>M7_ETM_IIDR3</td>
<td>NOOPEFLOW</td>
<td>NUMPROCOC12[3:0]</td>
<td>0x1D4</td>
</tr>
<tr>
<td>0x1D8</td>
<td>M7_ETM_IIDR1</td>
<td>TRCRARCHMAJ[3:0]</td>
<td>TRCRARCHMIN[3:0]</td>
<td>0x1D8</td>
<td>M7_ETM_IIDR1</td>
<td>TRCRARCHMAJ[3:0]</td>
<td>TRCRARCHMIN[3:0]</td>
<td>0x1D8</td>
</tr>
</tbody>
</table>
### Table 539. Cortex-M7 ETM register map and reset values (continued)

| Offset  | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x1F0   | M7_ETM_IDR4   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1F4   | M7_ETM_IDR5   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value | 1  | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 1  | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1F8 to 0x204 | Reserved    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x208   | M7_ETM_RSCTL2 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x20C   | M7_ETM_RSCTL3 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x210 to 0x27C | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x280   | M7_ETM_SSCC0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x284 to 0x29C | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2A0   | M7_ETM_SSCS0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | STATUS        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2A4 to 0x2BC | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2C0   | M7_ETM_SSPCIC0|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x2C4 to 0x30C | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x310   | M7_ETM_PDC   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x314   | M7_ETM_PDS   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x318 to 0x3F9C | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
### Table 539. Cortex-M7 ETM register map and reset values (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFA0</td>
<td>_CLAIMSET</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFA4</td>
<td>_CLAIMCLR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFA8 to 0xFAC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFB0</td>
<td>M7_ETM_LAR</td>
<td>ACCESS_W[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFB4</td>
<td>M7_ETM_LSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFB8</td>
<td>M7_ETM_AUTHSTAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFBC</td>
<td>M7_ETM_DEVARCH</td>
<td>ARCHITECT[10:0]</td>
<td>PRESENT</td>
<td>REVISION[3:0]</td>
<td>ARCHID[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 1 0 0 0</td>
<td>1 1 1 0 1 0 0 0 1 0 0 0</td>
<td></td>
<td>0 1 0 0 0 1 0</td>
<td>0 1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFC0 to 0xFC8</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFC8</td>
<td>M7_ETM_DEVTYPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD0</td>
<td>M7_ETM_PIDR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD4 to 0xFDC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>M7_ETM_PIDR0</td>
<td>PARTNUM[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 1 1 1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 64.6.6 Cortex-M7 cross trigger interface (CTI)

See Section 64.5.2.

### 64.7 References for debug infrastructure

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C
2. DDI 0480F (ID100313) - Arm® CoreSight SoC-400 r3p2 Technical Reference Manual, Issue G
3. DDI 0461B (ID010111) - Arm® CoreSight Trace Memory Controller r0p1 Technical Reference Manual, Issue B
5. DDI 0403D (ID100710) - Arm® v7-M Architecture Reference Manual, Issue E.b
6. DDI 0494-2a (ID062813) - Arm® CoreSight ETM-M7 r0p1 Technical Reference Manual, Issue D
65 Device electronic signature

The electronic signature is stored in the Flash memory area. It can be read using the JTAG/SWD or the CPU. It contains factory-programmed identification data that allow the user firmware or other external devices to automatically match its interface to the characteristics of the STM32H7A3/7B3/7B0xx microcontrollers.

65.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:
- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits can never be altered by the user.

The 96-bit unique device identifier can also be read in single bytes/half-words/words in different ways and then be concatenated using a custom algorithm.

**Base address: 0x08FF F800**

Address offset: 0x00
Read only = 0xXXXX XXXX where X is factory-programmed

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U_ID(31:0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **U_ID(31:0):** 31:0 unique ID bits

Address offset: 0x04
Read only = 0xXXXX XXXX where X is factory-programmed

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U_ID(63:48)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>U_ID(47:32)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **U_ID(63:32):** 63:32 unique ID bits
Address offset: 0x08
Read only = 0xXXXX XXXX where X is factory-programmed

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

U_ID(95:80)

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

U_ID(79:64)

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0  U_ID(95:64): 95:64 Unique ID bits.

### 65.2 Flash size

Base address: 0x08FF F80C
Address offset: 0x00
Read only = 0xXXXX where X is factory-programmed

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

F_SIZE

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:0  F_ID(15:0): Flash memory size
This bitfield indicates the size of the device Flash memory expressed in Kbytes.
As an example, 0x0400 corresponds to 1024 Kbytes.

### 65.3 Package data register

Base address: 0x08FF F80E
Address offset: 0x00
Read only = 0xXXXX where X is factory-programmed

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PKG[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 15:4  Reserved, must be kept at reset value.

Bits 3:0  **PKG[3:0]:** Package
- 0000: LQFP64
- 0001: TFBGA100/LQFP100
- 0010: LQFP100 devices with SMPS
- 0011: TFBGA100 devices with SMPS
- 0100: WLCSP132 devices with SMPS
- 0101: LQFP144
- 0110: LQFP144 devices with SMPS
- 0111: UFBGA169
- 1000: UFBGA176/LQFP176
- 1001: LQFP176 devices with SMPS
- 1010: UFBGA176 devices with SMPS
- 1100: TFBGA216
- 1110: TFBGA225
- Others: Reserved
## Revision history

### Table 540. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-Nov-2018</td>
<td>1</td>
<td>Initial release.</td>
</tr>
<tr>
<td>04-Dec-2018</td>
<td>2</td>
<td><strong>Section 12: System configuration controller (SYSCFG)</strong>&lt;br&gt;Added Section 12.3.1: Analog switch configuration management, Section 12.3.2: I2C Fm+ configuration, Section 12.3.4: Management of external interrupt line connections to GPIOs and Section 12.3.5: I/O speed in low-voltage mode.&lt;br&gt;SYSCFG peripheral mode configuration register (SYSCFG_PMCR):&lt;br&gt;– Updated reset value.&lt;br&gt;– Updated PBxFMP and I2CxFMP descriptions</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 38: True random number generator (RNG)</strong>&lt;br&gt;Corrected section numbering.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 43: General-purpose timers (TIM2/TIM3/TIM4/TIM5)</strong>&lt;br&gt;Removed all information related to BDTR register, MOE and OSSSI bits. Updated Figure 400: Output stage of Capture/Compare channel (channel 1).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 45: General-purpose timers (TIM15/TIM16/TIM17)</strong>&lt;br&gt;Changed COMDE of TIMx_DIER for TIM16/17 to reserved.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 52: Inter-integrated circuit (I2C) interface</strong>&lt;br&gt;Swapped I2C_SDA and I2C_SCL definition in Table 363: I2C input/output pins.</td>
</tr>
</tbody>
</table>
Table 540. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>28-Jan-2020</td>
<td>3</td>
<td>Changed document classification to public. Added support for STM32H7B0 Value line. Updated Figure 1: System architecture for STM32H7A3/7B3/7B0xx devices, Table 6: Register boundary addresses. Updated USART3 pins and added FDCAN1 in Section: Embedded bootloader.</td>
</tr>
</tbody>
</table>
Table 540. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>28-Jan-2020</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(continued)</td>
<td></td>
</tr>
</tbody>
</table>

**Section 6: Power control (PWR)**
- Added PDV_IN to Figure 18: Power control block diagram, and Table 30: PWR input/output signals connected to package pins or balls.
- Replaced RC48 by HSI48 in Figure 19: Power supply overview.
- In Section 6.4.4: Backup domain, removed PC1 from the list of functions available in VBAT mode when the VDD supply is absent and a supply is present on VBAT.
- Replaced PDR by POR/PDR in Section 6.5.1: Power-on reset (POR)/power-down reset (PDR).
- Removed reference to VBOR0 for BOR off in Section 6.5.2: Brownout reset (BOR).
- Updated Section 6.5.5: Battery voltage thresholds and Section 6.7.7: DStop and DStop2 modes.
- Added GFXSO bit (position 25) and update FLPS bit description in Section 6.8.1: PWR control register 1 (PWR_CR1).
- Changed VBATH/L bits to reserved in Section 6.8.3: PWR control register 2 (PWR_CR2).

**Section 7: Low-power SRD domain**
- Replaced LINUART1 by LPUART1.
- Updated Figure 39: Timing diagram of SRD SRAM-to-LPUART1 transfer with BDMA2 and SRD domain in Autonomous mode.
- Updated Section : EXTI programming.
- Updated DMAMUX2_C0CR value for DMAMUX2 SYNC0 in Table 48: BDMA2 and DMAMUX2 initialization sequence (DMAMUX2_INIT).
- Renamed Table 50 into “LPUART1 start programming.

**Section 8: Reset and clock control (RCC)**
- Replace rc48_ck by hsi48_ck in Figure 45: Top-level clock tree and Figure 59: Kernel clock distribution for USB (9), Updated Figure 46: HSE/LSE clock source and Figure 53: Kernel clock distribution for SAI, DFSDMs and SPDIFRX.
- Updated maximum VCO range in Figure 50: PLL block diagram, PLL3VCOSEL, PLL2VCOSEL and PLL1VCOSEL of RCC PLLs configuration register (RCC_PLLCFGR), in DIVN1/2/3[8:0] of RCC PLL1 dividers configuration register (RCC_PLL1DIVR)/RCC PLL2 dividers configuration register (RCC_PLL2DIVR)/RCC PLL3 dividers configuration register (RCC_PLL3DIVR) and FRACN1/2/3[12:0] of RCC PLL1 fractional divider register (RCC_PLL1FRACR)/RCC PLL2 fractional divider register (RCC_PLL2FRACR)/RCC PLL3 fractional divider register (RCC_PLL3FRACR).
- Changed sys_ck_cpu and sys_ck_bus max frequency to 280 MHz in Figure 52: Core and bus clock generation.
- In Table 58: Kernel clock distribution overview, updated DAC1. removed note 6 and changed DFSDM to DFSDM2 in note 7 related to SPI6 maximum frequency.
- Updated Figure 60: Kernel clock distribution For ADCs, SWP, RNG and FDCANs.
- Updated CPUCKRDY bit description in RCC source control register (RCC_CR).
- Updated TIMPRE bit description in RCC clock configuration register (RCC_CFGR).
- Replaced DCMIRST by DCMI_PSSIRST in RCC AHB2 peripheral reset register (RCC_AHB2RSTR), DCMEN by DCMI_PSSIEN in RCC AHB2 clock register (RCC_AHB2ENR), DCMIEN by DCMI_PSSISEN in RCC AHB2 sleep clock register (RCC_AHB2LPENR) and TMPSENSLPEN by DTSLPEN in RCC APB4 sleep clock register (RCC_APB4LPENR).
### Section 8: Reset and clock control (RCC) (continued)

Replaced TMPSENSRST by DTSRST in **RCC APB4 peripheral reset register (RCC_APB4RSTR)**, TMPSENSAMEN by DTSAMEN in **RCC SmartRun domain autonomous mode register (RCC_SRDAMR)** and TMPSENSEN by DTSEN in **RCC APB4 clock register (RCC_APB4ENR)**.

### Section 9: Clock recovery system (CRS)

Added **Section 9.3: CRS implementation**.

Updated **Section 9.8.1: CRS control register (CRS_CR) and Table 68: CRS register map and reset values**.

### Section 12: System configuration controller (SYSCFG)

Added **Section 12.3.1: Analog switch configuration management**, **Section 12.3.2: I2C Fm+ configuration**, **Section 12.3.4: Management of external interrupt line connections to GPIOs** and **Section 12.3.5: I/O speed in low-voltage mode**.

Updated reset value and PBxFMP and I2CxFMP descriptions in **SYSCFG peripheral mode configuration register (SYSCFG_PMCR)**.

Renamed **SYSCFG_BRK_LOCKUPR into SYSCFG_CFGCR**.

### Section 13: Block interconnect

Renamed DCMI to DCMI_PSSI in **Table 81: DMAMUX1, DMA1, DMA2 and BDMA1 connections**.

### Section 14: MDMA controller (MDMA)

Updated number of hardware trigger sources in **Section 14.2: MDMA main features**.

Updated **Figure 80: MDMA block diagram**.

### Section 16: Basic direct memory access controller (BDMA)

Added “Channel state and disabling channel” sub-part in **Section 16.4.5: BDMA channels**.

### Section 19: Nested vectored interrupt controller (NVIC)

Replaced DCMI by DCMI_PSSI and added cpu_fpu_it (position 81) in **Table 123: NVIC**.

### Section 20: Extended interrupt and event controller (EXTI)

Removed HSEM and Cortex-M7 SEV interrupts in **Table 126: EXTI Event input mapping**.

### Section 23: Flexible memory controller (FMC)

Replaced BCH8 by Hamming in **Section 23.8.6: Computation of the error correction code (ECC) in NAND Flash memory**

Updated **Figure 117: Mode D write access waveforms**.

Replaced FMC_CLK by fmc_ker_ck in the formulas of **Section : Wait management**.
### Table 540. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 28-Jan-2020  | 3        | **Section 24: Octo-SPI interface (OCTOSPI)**  
**Section 24.3.10: OCTOSPI Memory-mapped mode**: removed timeout from the list of events triggering BUSY fall; removed mention that Memory-mapped write operations are not supported on memories without write strobe or in Dual-quad mode in **Section: Memory-mapped mode configuration**.  
Updated **Section**: Sending the instruction only once (SIOO).  
Updated WRAP support subsection in **Section 24.3.6: Common functionality between the Regular-command and HyperBus modes**.  
Updated ABORT, EN and FTHRES[4:0] descriptions in **Section 24.5.1: OCTOSPI control register (OCTOSPI_CR)**, DLYBYP bit in **Section 24.5.2: OCTOSPI device configuration register 1 (OCTOSPI_DCR1)** and FLEVEL description in **Section 24.5.6: OCTOSPI status register (OCTOSPI_SR)**.  

**Section 25: OCTOSPI I/O manager (OCTOSPIM)**  
Updated **Section 25.3.3: OCTOSPIM multiplexer**.  

**Section 27: Analog-to-digital converters (ADC)**  
Removed VREF+ and VREF- ranges and added adc_sclk in **Table 185: ADC input/output pins**.  
Added reference to LDORDY bit in the whole section.  
Added LDO voltage regulator and replaced SMPPLUS control by Extended sample time option (SMPPLUS control) in **Section 27.3: ADC implementation**.  
Added adc_sclk in **Figure 154: ADC block diagram**.  
Updated **Section 27.4.3: ADC clocks**. Renamed adc_ker_ck into adc_ker_ck_inputs.  
Updated **Section**: BOOST control.  
Changed BOOSTE bit register to PWR_CR1 in **Section**: I/O analog switches voltage booster.  
Updated **Figure 156: ADC1 connectivity** and **Figure 157: ADC2 connectivity**.  
Changed TIMx_CCy into TIMx_OCy and all source names put in low-case in **Section 27.4.19: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)**.  
Updated **Section**: Single ADC operating modes support when oversampling to remove the mention that the offset correction is not supported in oversampling mode.  
Updated ADC input channel to which VBAT supply is connected in **Section 27.4.34: VBAT supply monitoring**.  
Updated RES[2:0] bitfield definition in **Section 27.6.4: ADC configuration register (ADC_CFRGR)**.  
Updated LSHIFT[3:0] “Shift left 14-bits” configuration in **Section 27.6.5: ADC configuration register 2 (ADC_CFRGR2)** register.  
Updated **Section 27.6.26: ADC calibration factors register (ADC_CALFACT)** register. Renamed VSENSEEN into TSEN in **Section 27.7.2: ADC x common control register (ADCx_CCR) (x=1/2)**.  
Replaced adc_hclk by adc_hclk in CKMODE[1:0] bitfield definition of **Section 27.7.2: ADC x common control register (ADCx_CCR) (x=1/2)**. |
### Table 540. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>28-Jan-2020</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

#### Section 28: Digital temperature sensor (DTS)
- Changed title to “Digital temperature sensor”.
- Removed all information related to quick measurement mode.
- Removed EN pin and Q_MEAS_opt from Figure 223: Temperature sensor functional block diagram.
- Updated Table 206: Trigger configuration.
- Updated formulas to calculate the temperature in Section 28.3.7: Temperature measurement principles.
- Removed startup time value from Section 28.3.10: On-off control and ready flag.
- Updated bitfield descriptions in Section 28.6.5: Temperature sensor data register (DTS_DR).
- Removed TS1_EN and updated TS1_INTRIG_SEL[3:0] to replace the list of triggers by a reference to Table Trigger configuration in Section 28.6.1: Temperature sensor configuration register 1 (DTS_CFG1).
- Changed DTS_OR bitfield names to uppercase (TS_opn into TS_OPn).

#### Section 29: Digital-to-analog converter (DAC)
- Updated Section 29.3: DAC implementation.
- Renamed internal signals:
  - Replaced sample and hold clock by dac_hold_ck.
  - Changed dac_chx_trg[0:15] into dac_chx_trg[1:15].
  - Updated TSELx bitfield description in DAC_CR register to add the correspondence between TSELx configurations and dac_chx_trgy.
- Updated Figure 227: Dual-channel DAC block diagram.
- Added Table 213: DAC1 interconnection and Table 214: DAC2 interconnection in Section 29.4.2: DAC pins and internal signals and replaced ck_lsi by lsi_ck.
- Updated Figure 230: Timing diagram for conversion with trigger disabled TEN = 0.
- Updated Section 29.4.4: Sample and hold mode to indicate that the lsi_ck/lse_ck (when available) must not be stopped when Sample and hold mode enabled.
- Removed Trigger selection tables from Section 29.4.7: DAC trigger selection.
- Updated supply voltages in Section 29.4.12: DAC channel buffer calibration.
- Updated CStop mode description for DAC2 in Section 29.5: DAC low-power modes.
- Updated Section 29.6: DAC interrupts.

#### Section 33: Digital filter for sigma delta modulators (DFSDM)
- Updated Section 33.7: DFSDM channel y registers (y=0..7).
- Updated Section 33.7.5: DFSDM channel y data input register (DFSDM_CHyDATINR).
- Updated Section 33.8: DFSDM filter x module registers (x=0..7).

#### Section 30: Voltage reference buffer (VREFBUF)
- Updated supported voltages in Section 30.2: VREFBUF functional description.

#### Section 34: Digital camera interface (DCMI)
- Updated pin names in Section 34.3: DCMI functional description.
<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>28-Jan-2020</td>
<td>3</td>
<td><strong>Section 38: True random number generator (RNG)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 38.1: Introduction and Section 38.2: RNG main features.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 38.4: RNG interrupts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Renamed NISTN by NISTC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 38.3.3: Random number generation and Figure 279: NIST SP800-90B entropy source model.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated note in Section 38.3.5: RNG operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 38.4: RNG interrupts, Section 38.5: RNG processing time, and Section 38.6: RNG entropy source validation and in particular Table 277: RNG configurations and Section 38.6.3: Data collection.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 38.7.3: RNG data register (RNG_DR).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added note in Section 38.7.4: RNG health test control register (RNG_HTCR).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 39: Cryptographic processor (CRYP)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated number of clock cycles for TDES in Section 39.2: CRYP main features.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Table 286: Key registers CRYP_KxR/LR endianness (TDES K1/2/3 and AES 128/192/256-bit keys).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 39.3.17: CRYP key registers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Table 286: Key registers CRYP_KxR/LR endianness (TDES K1/2/3 and AES 128/192/256-bit keys).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 39.4: CRYP interrupts</strong>: removed CRYP interrupt mapping diagram and updated Table 290: CRYP interrupt requests.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 39.5: CRYP processing time</strong>: updated Table 291: Processing latency for ECB, CBC and CTR and Table 292: Processing time (in clock cycle) for GCM and CCM per 128-bit block.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 40: Hash processor (HASH)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Removed MDMA feature.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Message size changed to $2^{64}-1$ in Section 40.1: Introduction and Section 40.4.3: About secure hash algorithms.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added Section 40.3: HASH implementation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simplified Figure 305: HASH block diagram and renamed hash_in_dma internal signal into hash_dma.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Figure 306: Message data swapping feature.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 40.4.8: HASH suspend/resume operations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Renamed the HASH_HRx registers located at address offset 0x0C into HASH_HRAx.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specified value returned by reading HASH data input register (HASH_DIN).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated ALGO description in HASH control register (HASH_CR).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 41: On-The-Fly Decryption engine - AXI (OTFDEC)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified Figure 309: Typical OTFDEC usage in the device.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replaced &quot;Flash memory&quot; with &quot;external read-only memory&quot; and of “SPI NOR Controller” with &quot;external memory controller in the whole document.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified Section 41.5.3: Encrypting for OTFDEC.</td>
</tr>
</tbody>
</table>
Table 540. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>26-Jan-2020</td>
<td>3</td>
<td>(continued)</td>
</tr>
</tbody>
</table>

**Section 42: Advanced-control timers (TIM1/TIM8)**
- Updated Figure 313: Advanced-control timer block diagram.
- Updated Section 42.3.3: Repetition counter.
- Updated Section 42.3.16: Using the break function and Figure 356: Break and Break2 circuitry overview.
- Updated Section 42.3.29: Debug mode
- Updated Section 42.4.2: TIMx control register 2 (TIMx_CR2)\( (x = 1, 8) \)
- Aligned TS\[2:0\] field in Section 42.4.3: TIMx slave mode control register (TIMx_SMCR)\( (x = 1, 8) \)
- Updated Section 42.4.5: TIMx status register (TIMx_SR)\( (x = 1, 8) \), Section 42.4.6: TIMx event generation register (TIMx_EGR)\( (x = 1, 8) \). Updated Table 307: TIM1 register map and reset values and Table 308: TIM8 register map and reset values.

**Section 43: General-purpose timers (TIM2/TIM3/TIM4/TIM5)**
- Updated Figure 373: General-purpose timer block diagram.
- Updated Section : External clock source mode 2.
- Updated Figure 400: Output stage of Capture/Compare channel (channel 1).
- Updated Section 43.4.2: TIMx control register 2 (TIMx_CR2)\( (x = 2 to 5) \), Section 43.4.5: TIMx status register (TIMx_SR)\( (x = 2 to 5) \), Section 43.4.11: TIMx capture/compare enable register (TIMx_CCER)\( (x = 2 to 5) \), Section 43.4.12: TIMx counter [alternate] (TIMx_CNT)\( (x = 2 to 5) \) and Table 312: TIM2/TIM3/TIM4/TIM5 register map and reset values.

**Section 44: General-purpose timers (TIM12/TIM13/TIM14)**
- Updated Figure 422: General-purpose timer block diagram (TIM12).
- Updated Figure 436: Capture/compare channel 1 main circuit and Figure 437: Output stage of capture/compare channel (channel 1)
- Updated Section 44.4.2: TIM12 control register 2 (TIM12_CR2), Section 44.4.5: TIM12 status register (TIM12_SR) and Section 44.4.6: TIM12 event generation register (TIM12_EGR), Updated OC1FE description in Section 44.4.7: TIM12 capture/compare mode register 1 [alternate] (TIM12_CCMR1) and Section 44.5.5: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)\( (x = 13 to 14) \).

**Section 45: General-purpose timers (TIM15/TIM16/TIM17)**
- Updated Figure 448: TIM15 block diagram and Figure 449: TIM16/TIM17 block diagram and Figure 463: Capture/compare channel 1 main circuit.
- Updated source for break channels in Section 45.4.13: Using the break function.
- Updated Figure 473: Break circuitry overview.
- Updated Section 45.5.2: TIM15 control register 2 (TIM15_CR2), Section 45.5.5: TIM15 status register (TIM15_SR), Section 45.5.9: TIM15 capture/compare enable register (TIM15_CCER), Section 45.6.2: TIMx control register 2 (TIMx_CR2)\( (x = 16 to 17) \), Section 45.6.4: TIMx status register (TIMx_SR)\( (x = 16 to 17) \), Section 45.6.6: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)\( (x = 16 to 17) \) and Section 45.6.8: TIMx capture/compare enable register (TIMx_CCER)\( (x = 16 to 17) \).
- Replaced DT\[7:0\] by DTG\[7:0\] in Table 321: TIM15 register map and reset values and Table 323: TIM16/TIM17 register map and reset values. Updated OC1FE description in Section 45.5.8: TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) and Section 45.6.6: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)\( (x = 16 to 17) \).
### Table 540. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>28-Jan-2020</td>
<td>3</td>
<td><strong>Section 47: Low-power timer (LPTIM)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 47.4.5: Glitch filter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 51: Tamper and backup registers (TAMP)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Figure 504: TAMP block diagram. Updated Table 355: TAMP internal input/output signals and Table 356: TAMP interconnection. Updated Section : TAMP backup registers. Added: Table 359: TAMP pins functionality over modes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 52: Inter-integrated circuit (I2C) interface</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 52.3: I2C implementation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 52.6: I2C interrupts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 52.7.2: I2C control register 2 (I2C_CR2), Section 52.7.3: I2C own address 1 register (I2C_OAR1) and Section 52.7.8: I2C interrupt clear register (I2C_ICR).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 53: Universal synchronous/asynchronous receiver transmitter (USART/UART)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 53.5.4: USART FIFOs and thresholds.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed USART_TDR into USART_RDR in Figure 557: Reception using DMA.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated RTO bitfield description in Section 53.7.7: USART receiver timeout register (USART_RTOR) register. Changed reset value for FIFO enabled and disabled of Section 53.7.9: USART interrupt and status register [alternate] (USART_ISR) and Table 386: USART register map and reset values. Changed reset value of USART receive data register (USART_RDR) and USART transmit data register (USART_TDR). Changed reset value for FIFO enabled and disabled of Section 54.6.9: LPUART interrupt flag clear register (LPUART_ICR) and Table 393: LPUART register map and reset values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 54: Low-power universal asynchronous receiver transmitter (LPUART)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added Section 54.3: LPUART implementation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed LPUART_TDR into LPUART_RDR in Figure 571: Reception using DMA.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 54.4.4: LPUART FIFOs and thresholds.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 55: Serial peripheral interface (SPI)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Table 394: STM32H7xx SPI features.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated note below Figure 581: Master and three independent slaves at star topology. Updated Figure 592: Waveform examples. Updated Section 55.11.1: SPI/I2S control register 1 (SPI2S_CR1). Updated CRCSIZE[4:0] description in Section 55.11.3: SPI configuration register 1 (SPI_CFG1).</td>
</tr>
</tbody>
</table>
### Table 540. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>28-Jan-2020</td>
<td>3</td>
<td><strong>Section 58: Single Wire Protocol Master Interface (SWPMI)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Table 422: SWPMI input/output signals connected to package pins or balls.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replaced SWPSCR of RCC_D2CCIP1R by SWPSEL of RCC_CDCCIP1R in Section 58.3.1: SWPMI block diagram.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 56: Serial audio interface (SAI)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added Section 56.3: SAI implementation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of Dn and CKn lines made generic in Figure 611: SAI functional block diagram.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added note to indicate that all Dn and CKn might not be available on all SAI instances below Figure 611: SAI functional block diagram, Table 404: SAI input/output pins.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Figure 620: Start-up sequence and added note related to the fact that F_s and F_SCK_x formulas are valid only if NODIV = 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 60: Secure digital input/output MultiMediaCard interface (SDMMC)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added Section 60.2: SDMMC implementation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Section 61: Controller area network with flexible data rate (FDCAN)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 61.1: Introduction, Section 61.2: FDCAN main features, Added Section 61.3: FDCAN implementation. Added Table 460: CAN triggers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Figure 699: FDCAN block diagram to remove FDCAN1/2_T/RXF_D_MODE signals.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section : Acceptance filter, Section 61.4.4: Bit timing, Section 61.4.5: Clock calibration on CAN, Section 61.4.6: Application, and Section : Timing of interface signals.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Table 481: Standard message ID filter element, Table 483: Extended message ID filter element, Table 486: Trigger memory element description, Figure 704: Standard message ID filter path and Figure 705: Extended message ID filter path.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Table 481: Standard message ID filter element, Table 483: Extended message ID filter element, Table 486: Trigger memory element description.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Section 61.5.6: FDCAN CC control register (FDCAN_CCCR), Section 61.5.7: FDCAN nominal bit timing and prescaler register (FDCAN_NBTP), Section 61.5.12: FDCAN error counter register (FDCAN_ECR), Section 61.5.15: FDCAN interrupt register (FDCAN_IR), Section 61.5.18: FDCAN interrupt line enable register (FDCAN_ILE), Section 61.5.21: FDCAN extended ID filter configuration register (FDCAN_XIDFC), Section 61.5.37: FDCAN Tx buffer request pending register (FDCAN_TXBRP), Section 61.5.52: FDCAN TT operation control register (FDCAN_TTOCN) and Section 61.5.64: FDCAN TT trigger select register (FDCAN_TTTS).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated Table 487: FDCAN register map and reset values and Table 488: CCU register map and reset values.</td>
</tr>
</tbody>
</table>
Table 540. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>28-Jan-2020</td>
<td>3</td>
<td><strong>Section 62: USB on-the-go high-speed (OTG_HS)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.2.1: General features.</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.14.2: OTG interrupt register (OTG_GOTGINT).</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.14.5: OTG reset register (OTG_GRSTCTL).</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.14.6: OTG core interrupt register (OTG_GINTSTS).</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.14.18: OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ).</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.14.22: OTG host frame interval register (OTG_HFIR).</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.14.33: OTG host channel x transfer size register (OTG_HCTSIZx).</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.14.39: OTG device configuration register (OTG_DCFG).</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.14.40: OTG device control register (OTG_DCTL).</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.14.41: OTG device status register (OTG_DSTS).</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.14.42: OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK).</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.14.43: OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK).</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.14.44: OTG device IN endpoint x control register (OTG_DIEPCTLx).</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.14.45: OTG device OUT endpoint x interrupt register (OTG_DOEPINTx).</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.15.3: Device initialization and Section 62.15.6: Device programming model.</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated <a href="#">Section 62.10: OTG_HS Dynamic update of the OTG_HFIR register.</a></td>
</tr>
</tbody>
</table>

**Section 64: Debug infrastructure**

Updated [ETM CoreSight device architecture register (M7_ETM_DEVARCH).](#)

Added revision Z in [DBGMCU identity code register (DBGMCU_IDC).](#)
# Index

## A

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_AWD2CR</td>
<td>1039</td>
</tr>
<tr>
<td>ADC_AWD3CR</td>
<td>1040</td>
</tr>
<tr>
<td>ADC_CALFACT</td>
<td>1043</td>
</tr>
<tr>
<td>ADC_CALFACT2</td>
<td>1043</td>
</tr>
<tr>
<td>ADC_CFGR</td>
<td>1021</td>
</tr>
<tr>
<td>ADC_CFGR2</td>
<td>1025</td>
</tr>
<tr>
<td>ADC_CR</td>
<td>1016</td>
</tr>
<tr>
<td>ADC_DIFSEL</td>
<td>1042</td>
</tr>
<tr>
<td>ADC_DR</td>
<td>1035</td>
</tr>
<tr>
<td>ADC_HTR1</td>
<td>1030</td>
</tr>
<tr>
<td>ADC_HTR2</td>
<td>1041</td>
</tr>
<tr>
<td>ADC_HTR3</td>
<td>1042</td>
</tr>
<tr>
<td>ADC_IER</td>
<td>1014</td>
</tr>
<tr>
<td>ADC_ISR</td>
<td>1010</td>
</tr>
<tr>
<td>ADC_JDRy</td>
<td>1039</td>
</tr>
<tr>
<td>ADC_JSQRY</td>
<td>1036</td>
</tr>
<tr>
<td>ADC_LTR1</td>
<td>1029</td>
</tr>
<tr>
<td>ADC_LTR2</td>
<td>1040</td>
</tr>
<tr>
<td>ADC_LTR3</td>
<td>1041</td>
</tr>
<tr>
<td>ADC_OFRy</td>
<td>1038</td>
</tr>
<tr>
<td>ADC_PCSELY</td>
<td>1029</td>
</tr>
<tr>
<td>ADC_SMPR1</td>
<td>1027</td>
</tr>
<tr>
<td>ADC_SMPR2</td>
<td>1028</td>
</tr>
<tr>
<td>ADC_SQR1</td>
<td>1031</td>
</tr>
<tr>
<td>ADC_SQR2</td>
<td>1032</td>
</tr>
<tr>
<td>ADC_SQR3</td>
<td>1033</td>
</tr>
<tr>
<td>ADC_SQR4</td>
<td>1034</td>
</tr>
<tr>
<td>ADCx_CCR</td>
<td>1046</td>
</tr>
<tr>
<td>ADCx_CDR</td>
<td>1049</td>
</tr>
<tr>
<td>ADCx_CDR2</td>
<td>1049</td>
</tr>
<tr>
<td>ADCx_CSR</td>
<td>1044</td>
</tr>
<tr>
<td>AP_BASE</td>
<td>2756</td>
</tr>
<tr>
<td>AP_CSW</td>
<td>2754</td>
</tr>
<tr>
<td>AP_IDR</td>
<td>2756</td>
</tr>
<tr>
<td>AXI_COMP_ID_0</td>
<td>110</td>
</tr>
<tr>
<td>AXI_COMP_ID_1</td>
<td>111</td>
</tr>
<tr>
<td>AXI_COMP_ID_2</td>
<td>111</td>
</tr>
<tr>
<td>AXI_COMP_ID_3</td>
<td>112</td>
</tr>
<tr>
<td>AXI_INIx_FN_MOD</td>
<td>116</td>
</tr>
<tr>
<td>AXI_INIx_FN_MOD_AHB</td>
<td>115</td>
</tr>
<tr>
<td>AXI_INIx_FN_MOD2</td>
<td>114</td>
</tr>
<tr>
<td>AXI_INIx_READ_QOS</td>
<td>115</td>
</tr>
<tr>
<td>AXI_INIx_WRITE_QOS</td>
<td>116</td>
</tr>
<tr>
<td>AXI_PERIPH_ID_0</td>
<td>108</td>
</tr>
<tr>
<td>AXI_PERIPH_ID_1</td>
<td>109</td>
</tr>
<tr>
<td>AXI_PERIPH_ID_2</td>
<td>109</td>
</tr>
<tr>
<td>AXI_PERIPH_ID_3</td>
<td>110</td>
</tr>
<tr>
<td>AXI_PERIPH_ID_4</td>
<td>108</td>
</tr>
<tr>
<td>AXI_TARGx_FN_MOD</td>
<td>114</td>
</tr>
<tr>
<td>AXI_TARGx_FN_MOD_ISS_BM</td>
<td>112</td>
</tr>
<tr>
<td>AXI_TARGx_FN_MOD_LA</td>
<td>113</td>
</tr>
<tr>
<td>AXI_TARGx_FN_MOD2</td>
<td>113</td>
</tr>
<tr>
<td>BDMA_CCRx</td>
<td>643</td>
</tr>
<tr>
<td>BDMA_CM0ARx</td>
<td>648</td>
</tr>
<tr>
<td>BDMA_CM1ARx</td>
<td>649</td>
</tr>
<tr>
<td>BDMA_CNDTRx</td>
<td>647</td>
</tr>
<tr>
<td>BDMA_CPARx</td>
<td>647</td>
</tr>
<tr>
<td>BDMA_IFCR</td>
<td>642</td>
</tr>
<tr>
<td>BDMA_ISR</td>
<td>639</td>
</tr>
<tr>
<td>CEC_CFGR</td>
<td>2725</td>
</tr>
<tr>
<td>CEC_CR</td>
<td>2724</td>
</tr>
<tr>
<td>CEC_IER</td>
<td>2729</td>
</tr>
<tr>
<td>CEC_ISR</td>
<td>2727</td>
</tr>
<tr>
<td>CEC_RXDR</td>
<td>2727</td>
</tr>
<tr>
<td>CEC_TXDR</td>
<td>2727</td>
</tr>
<tr>
<td>COMP_CFG1</td>
<td>1124</td>
</tr>
<tr>
<td>COMP_CFG2</td>
<td>1126</td>
</tr>
<tr>
<td>COMP_ICFR</td>
<td>1122</td>
</tr>
<tr>
<td>COMP_OR</td>
<td>1123</td>
</tr>
<tr>
<td>COMP_SR</td>
<td>1122</td>
</tr>
<tr>
<td>CRC_CR</td>
<td>777</td>
</tr>
<tr>
<td>CRC_DR</td>
<td>776</td>
</tr>
<tr>
<td>CRC_IDR</td>
<td>776</td>
</tr>
<tr>
<td>CRC_INIT</td>
<td>777</td>
</tr>
<tr>
<td>CRC_POL</td>
<td>778</td>
</tr>
<tr>
<td>CRS_CFG1</td>
<td>484</td>
</tr>
<tr>
<td>CRS_CR</td>
<td>483</td>
</tr>
<tr>
<td>CRS_ICR</td>
<td>487</td>
</tr>
<tr>
<td>CRS_ISR</td>
<td>485</td>
</tr>
<tr>
<td>CRYPT_CR</td>
<td>1377</td>
</tr>
<tr>
<td>CRYPT_CSGCMCCMxR</td>
<td>1388</td>
</tr>
<tr>
<td>CRYPT_CSGCMxR</td>
<td>1389</td>
</tr>
<tr>
<td>CRYPT_DIN</td>
<td>1379</td>
</tr>
<tr>
<td>CRYPT_DMACR</td>
<td>1381</td>
</tr>
<tr>
<td>CRYPT_DOUT</td>
<td>1380</td>
</tr>
<tr>
<td>CRYPT_IMSCR</td>
<td>1381</td>
</tr>
<tr>
<td>CRYPT_IV0LR</td>
<td>1387</td>
</tr>
<tr>
<td>CRYPT_IV0RR</td>
<td>1387</td>
</tr>
<tr>
<td>CRYPT_IV1LR</td>
<td>1388</td>
</tr>
<tr>
<td>CRYPT_IV1RR</td>
<td>1388</td>
</tr>
<tr>
<td>CRYPT_K0LR</td>
<td>1383</td>
</tr>
<tr>
<td>CRYPT_K0RR</td>
<td>1384</td>
</tr>
<tr>
<td>CRYPT_K1LR</td>
<td>1384</td>
</tr>
<tr>
<td>CRYPT_K1RR</td>
<td>1385</td>
</tr>
<tr>
<td>Index</td>
<td>RM0455</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------</td>
</tr>
<tr>
<td>CRYP_K2LR</td>
<td>1385</td>
</tr>
<tr>
<td>CRYP_K2RR</td>
<td>1385</td>
</tr>
<tr>
<td>CRYP_K3LR</td>
<td>1386</td>
</tr>
<tr>
<td>CRYP_K3RR</td>
<td>1386</td>
</tr>
<tr>
<td>CRYP_MISR</td>
<td>1382</td>
</tr>
<tr>
<td>CRYP_RISR</td>
<td>1382</td>
</tr>
<tr>
<td>CRYP_SR</td>
<td>1379</td>
</tr>
<tr>
<td>CSTF_AUTHSTAT</td>
<td>2789</td>
</tr>
<tr>
<td>CSTF_CIDR0</td>
<td>2793</td>
</tr>
<tr>
<td>CSTF_CIDR1</td>
<td>2793</td>
</tr>
<tr>
<td>CSTF_CIDR2</td>
<td>2794</td>
</tr>
<tr>
<td>CSTF_CIDR3</td>
<td>2794</td>
</tr>
<tr>
<td>CSTF_CLAIMCLR</td>
<td>2788</td>
</tr>
<tr>
<td>CSTF_CLAIMSET</td>
<td>2787</td>
</tr>
<tr>
<td>CSTF_CTRL</td>
<td>2786</td>
</tr>
<tr>
<td>CSTF_DEVID</td>
<td>2790</td>
</tr>
<tr>
<td>CSTF_DEVTYPE</td>
<td>2790</td>
</tr>
<tr>
<td>CSTF_LAR</td>
<td>2788</td>
</tr>
<tr>
<td>CSTF_LSR</td>
<td>2789</td>
</tr>
<tr>
<td>CSTF_PIDR0</td>
<td>2791</td>
</tr>
<tr>
<td>CSTF_PIDR1</td>
<td>2792</td>
</tr>
<tr>
<td>CSTF_PIDR2</td>
<td>2792</td>
</tr>
<tr>
<td>CSTF_PIDR3</td>
<td>2793</td>
</tr>
<tr>
<td>CSTF_PIDR4</td>
<td>2791</td>
</tr>
<tr>
<td>CSTF_PRIORITY</td>
<td>2787</td>
</tr>
<tr>
<td>CTI_APPCLEAR</td>
<td>2770</td>
</tr>
<tr>
<td>CTI_APPPULSE</td>
<td>2771</td>
</tr>
<tr>
<td>CTI_APPSET</td>
<td>2770</td>
</tr>
<tr>
<td>CTI_AUTHSTAT</td>
<td>2777</td>
</tr>
<tr>
<td>CTI_CHINSTS</td>
<td>2774</td>
</tr>
<tr>
<td>CTI_CHOOUTSTS</td>
<td>2774</td>
</tr>
<tr>
<td>CTI_CIDR0</td>
<td>2781</td>
</tr>
<tr>
<td>CTI_CIDR1</td>
<td>2781</td>
</tr>
<tr>
<td>CTI_CIDR2</td>
<td>2782</td>
</tr>
<tr>
<td>CTI_CIDR3</td>
<td>2782</td>
</tr>
<tr>
<td>CTI_CLAIMCLR</td>
<td>2776</td>
</tr>
<tr>
<td>CTI_CLAIMSET</td>
<td>2775</td>
</tr>
<tr>
<td>CTI_CONTROL</td>
<td>2769</td>
</tr>
<tr>
<td>CTI_DEVID</td>
<td>2778</td>
</tr>
<tr>
<td>CTI_DEVTYPE</td>
<td>2778</td>
</tr>
<tr>
<td>CTI_GATE</td>
<td>2775</td>
</tr>
<tr>
<td>CTI_INENx</td>
<td>2771</td>
</tr>
<tr>
<td>CTI_INTACK</td>
<td>2769</td>
</tr>
<tr>
<td>CTI_LAR</td>
<td>2776</td>
</tr>
<tr>
<td>CTI_LSR</td>
<td>2777</td>
</tr>
<tr>
<td>CTI_OUTENx</td>
<td>2772</td>
</tr>
<tr>
<td>CTI_PIDR0</td>
<td>2779</td>
</tr>
<tr>
<td>CTI_PIDR1</td>
<td>2780</td>
</tr>
<tr>
<td>CTI_PIDR2</td>
<td>2780</td>
</tr>
<tr>
<td>CTI_PIDR3</td>
<td>2781</td>
</tr>
<tr>
<td>CTI_PIDR4</td>
<td>2779</td>
</tr>
<tr>
<td>CTI_TRGISTS</td>
<td>2773</td>
</tr>
<tr>
<td>Index</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>2773</td>
<td>CTI_TRGOSTS</td>
</tr>
<tr>
<td>1102</td>
<td>DAC_CCR</td>
</tr>
<tr>
<td>1091</td>
<td>DAC_CR</td>
</tr>
<tr>
<td>1095</td>
<td>DAC_DHR12L1</td>
</tr>
<tr>
<td>1097</td>
<td>DAC_DHR12L2</td>
</tr>
<tr>
<td>1098</td>
<td>DAC_DHR12LD</td>
</tr>
<tr>
<td>1095</td>
<td>DAC_DHR12R1</td>
</tr>
<tr>
<td>1096</td>
<td>DAC_DHR12R2</td>
</tr>
<tr>
<td>1098</td>
<td>DAC_DHR12RD</td>
</tr>
<tr>
<td>1096</td>
<td>DAC_DHR8R1</td>
</tr>
<tr>
<td>1097</td>
<td>DAC_DHR8R2</td>
</tr>
<tr>
<td>1099</td>
<td>DAC_DHR8RD</td>
</tr>
<tr>
<td>1099</td>
<td>DAC_DOR1</td>
</tr>
<tr>
<td>1100</td>
<td>DAC_DOR2</td>
</tr>
<tr>
<td>1102</td>
<td>DAC_MCR</td>
</tr>
<tr>
<td>1105</td>
<td>DAC_SHHR</td>
</tr>
<tr>
<td>1105</td>
<td>DAC_SHRR</td>
</tr>
<tr>
<td>1104</td>
<td>DAC_SHSR1</td>
</tr>
<tr>
<td>1104</td>
<td>DAC_SHSR2</td>
</tr>
<tr>
<td>1100</td>
<td>DAC_SR</td>
</tr>
<tr>
<td>1094</td>
<td>DAC_SWTRGR</td>
</tr>
<tr>
<td>2854</td>
<td>DBGMCU_APB1LFZ1</td>
</tr>
<tr>
<td>2855</td>
<td>DBGMCU_APB2FZ1</td>
</tr>
<tr>
<td>2853</td>
<td>DBGMCU_APB3FZ1</td>
</tr>
<tr>
<td>2856</td>
<td>DBGMCU_APB4FZ1</td>
</tr>
<tr>
<td>2852</td>
<td>DBGMCU_CR</td>
</tr>
<tr>
<td>2851</td>
<td>DBGMCU_IDC</td>
</tr>
<tr>
<td>1226</td>
<td>DCMI_CR</td>
</tr>
<tr>
<td>1235</td>
<td>DCMI_CWSIZE</td>
</tr>
<tr>
<td>1234</td>
<td>DCMI_CWSTRRT</td>
</tr>
<tr>
<td>1235</td>
<td>DCMI_DR</td>
</tr>
<tr>
<td>1232</td>
<td>DCMI_ESCR</td>
</tr>
<tr>
<td>1234</td>
<td>DCMI_ESUR</td>
</tr>
<tr>
<td>1232</td>
<td>DCMI_ICR</td>
</tr>
<tr>
<td>1230</td>
<td>DCMI_IER</td>
</tr>
<tr>
<td>1231</td>
<td>DCMI_MIS</td>
</tr>
<tr>
<td>1229</td>
<td>DCMI_RIS</td>
</tr>
<tr>
<td>1228</td>
<td>DCMI_SR</td>
</tr>
<tr>
<td>1182</td>
<td>DFSDM_CHyAWSCDR</td>
</tr>
<tr>
<td>1179</td>
<td>DFSDM_CHyCFGR1</td>
</tr>
<tr>
<td>1181</td>
<td>DFSDM_CHyCFGR2</td>
</tr>
<tr>
<td>1183</td>
<td>DFSDM_CHyDATINR</td>
</tr>
<tr>
<td>1184</td>
<td>DFSDM_CHyDLYR</td>
</tr>
<tr>
<td>1183</td>
<td>DFSDM_CHyWDATR</td>
</tr>
<tr>
<td>1197</td>
<td>DFSDMFLTxAWCFR</td>
</tr>
<tr>
<td>1195</td>
<td>DFSDMFLTxAWHTR</td>
</tr>
<tr>
<td>1195</td>
<td>DFSDMFLTxAWLTR</td>
</tr>
<tr>
<td>1196</td>
<td>DFSDMFLTxAWSR</td>
</tr>
<tr>
<td>1198</td>
<td>DFSDMFLTxCNVTIMR</td>
</tr>
<tr>
<td>Index</td>
<td>RM0455</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>DFSDM_FLTxC1</td>
<td>1185</td>
</tr>
<tr>
<td>DFSDM_FLTxC2</td>
<td>1188</td>
</tr>
<tr>
<td>DFSDM_FLTxE</td>
<td>1197</td>
</tr>
<tr>
<td>DFSDM_FLTXE</td>
<td>1198</td>
</tr>
<tr>
<td>DFSDM_FLTxF</td>
<td>1192</td>
</tr>
<tr>
<td>DFSDM_FLTxG</td>
<td>1191</td>
</tr>
<tr>
<td>DFSDM_FLTxH</td>
<td>1189</td>
</tr>
<tr>
<td>DFSDM_FLTxI</td>
<td>1192</td>
</tr>
<tr>
<td>DFSDM_FLTxJ</td>
<td>1183</td>
</tr>
<tr>
<td>DFSDM_FLTxK</td>
<td>1194</td>
</tr>
<tr>
<td>DLYB_CFGR</td>
<td>927</td>
</tr>
<tr>
<td>DLYB_CR</td>
<td>927</td>
</tr>
<tr>
<td>DMA_HIFCR</td>
<td>615</td>
</tr>
<tr>
<td>DMA_HISR</td>
<td>614</td>
</tr>
<tr>
<td>DMA_LIFCR</td>
<td>615</td>
</tr>
<tr>
<td>DMA_LISR</td>
<td>613</td>
</tr>
<tr>
<td>DMA_SxCR</td>
<td>616</td>
</tr>
<tr>
<td>DMA_SxF</td>
<td>621</td>
</tr>
<tr>
<td>DMA_SxM0AR</td>
<td>620</td>
</tr>
<tr>
<td>DMA_SxM1AR</td>
<td>621</td>
</tr>
<tr>
<td>DMA_SxNDTR</td>
<td>619</td>
</tr>
<tr>
<td>DMA_SxPAR</td>
<td>620</td>
</tr>
<tr>
<td>DMA2D_AMTCCR</td>
<td>708</td>
</tr>
<tr>
<td>DMA2D_BGCMA</td>
<td>703</td>
</tr>
<tr>
<td>DMA2D_BGCOLR</td>
<td>702</td>
</tr>
<tr>
<td>DMA2D_BGMAR</td>
<td>696</td>
</tr>
<tr>
<td>DMA2D_BGOR</td>
<td>696</td>
</tr>
<tr>
<td>DMA2D_BGPFCR</td>
<td>700</td>
</tr>
<tr>
<td>DMA2D_BOR</td>
<td>691</td>
</tr>
<tr>
<td>DMA2D_FGCMAR</td>
<td>702</td>
</tr>
<tr>
<td>DMA2D_FGCOLR</td>
<td>699</td>
</tr>
<tr>
<td>DMA2D_FGMAR</td>
<td>695</td>
</tr>
<tr>
<td>DMA2D_FGOR</td>
<td>695</td>
</tr>
<tr>
<td>DMA2D_FGPFCCR</td>
<td>697</td>
</tr>
<tr>
<td>DMA2D_IFCR</td>
<td>694</td>
</tr>
<tr>
<td>DMA2D_ISR</td>
<td>693</td>
</tr>
<tr>
<td>DMA2D_ILR</td>
<td>707</td>
</tr>
<tr>
<td>DMA2D_NLR</td>
<td>707</td>
</tr>
<tr>
<td>DMA2D_OCOLR</td>
<td>704</td>
</tr>
<tr>
<td>DMA2D_OMAR</td>
<td>706</td>
</tr>
<tr>
<td>DMA2D_OOR</td>
<td>706</td>
</tr>
<tr>
<td>DMA2D_OPFCCR</td>
<td>703</td>
</tr>
<tr>
<td>DMAMUX_CxCR</td>
<td>664</td>
</tr>
<tr>
<td>DMAMUX_RGCFR</td>
<td>671</td>
</tr>
<tr>
<td>DMAMUX1_CFR</td>
<td>667</td>
</tr>
<tr>
<td>DMAMUX1_CSR</td>
<td>666</td>
</tr>
<tr>
<td>DMAMUX1_CxCR</td>
<td>664</td>
</tr>
<tr>
<td>DMAMUX1_RGCFR</td>
<td>671</td>
</tr>
<tr>
<td>DMAMUX1_RGSR</td>
<td>670</td>
</tr>
<tr>
<td>DMAMUX1_RGxCR</td>
<td>668</td>
</tr>
<tr>
<td>DMAMUX2_CFR</td>
<td>667</td>
</tr>
<tr>
<td>DMAMUX2_CSR</td>
<td>666</td>
</tr>
<tr>
<td>Symbol</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------</td>
</tr>
<tr>
<td>DMAMUX2_CxCR</td>
<td>665</td>
</tr>
<tr>
<td>DMAMUX2_RGCFR</td>
<td>671</td>
</tr>
<tr>
<td>DMAMUX2_RGSR</td>
<td>670</td>
</tr>
<tr>
<td>DMAMUX2_RGxCR</td>
<td>668</td>
</tr>
<tr>
<td>DP_ABORT</td>
<td>2745</td>
</tr>
<tr>
<td>DP_CTRL/STAT</td>
<td>2746</td>
</tr>
<tr>
<td>DP_DLCR</td>
<td>2748</td>
</tr>
<tr>
<td>DP_DLPIDR</td>
<td>2749</td>
</tr>
<tr>
<td>DP_DPIDR</td>
<td>2744</td>
</tr>
<tr>
<td>DP_RDBUFF</td>
<td>2751</td>
</tr>
<tr>
<td>DP_RESEND</td>
<td>2750</td>
</tr>
<tr>
<td>DP_SELECT</td>
<td>2750</td>
</tr>
<tr>
<td>DP_TARGETID</td>
<td>2749</td>
</tr>
<tr>
<td>DP_TARGETSEL</td>
<td>2751</td>
</tr>
<tr>
<td>DTS_CFRG1</td>
<td>1063</td>
</tr>
<tr>
<td>DTS_DR</td>
<td>1065</td>
</tr>
<tr>
<td>DTS_ICIFR</td>
<td>1068</td>
</tr>
<tr>
<td>DTS_ITENR</td>
<td>1067</td>
</tr>
<tr>
<td>DTS_ITR1</td>
<td>1065</td>
</tr>
<tr>
<td>DTS_OR</td>
<td>1069</td>
</tr>
<tr>
<td>DTS_RAMPVALR</td>
<td>1064</td>
</tr>
<tr>
<td>DTS_SR</td>
<td>1066</td>
</tr>
<tr>
<td>DTS_T0VALR1</td>
<td>1064</td>
</tr>
<tr>
<td>ETF_AUTHSTAT</td>
<td>2811</td>
</tr>
<tr>
<td>ETF_BUFWM</td>
<td>2805</td>
</tr>
<tr>
<td>ETF_CBUFLVL</td>
<td>2805</td>
</tr>
<tr>
<td>ETF_CIDR0</td>
<td>2815</td>
</tr>
<tr>
<td>ETF_CIDR1</td>
<td>2816</td>
</tr>
<tr>
<td>ETF_CIDR2</td>
<td>2816</td>
</tr>
<tr>
<td>ETF_CIDR3</td>
<td>2816</td>
</tr>
<tr>
<td>ETF_CLAIMCLR</td>
<td>2809</td>
</tr>
<tr>
<td>ETF_CLAIMSET</td>
<td>2809</td>
</tr>
<tr>
<td>ETF_CTL</td>
<td>2802</td>
</tr>
<tr>
<td>ETF_DEVID</td>
<td>2812</td>
</tr>
<tr>
<td>ETF_DEVTYPE</td>
<td>2812</td>
</tr>
<tr>
<td>ETF_FFCR</td>
<td>2806</td>
</tr>
<tr>
<td>ETF_FFSR</td>
<td>2806</td>
</tr>
<tr>
<td>ETF_LAR</td>
<td>2810</td>
</tr>
<tr>
<td>ETF_LBUFLVL</td>
<td>2804</td>
</tr>
<tr>
<td>ETF_LSR</td>
<td>2810</td>
</tr>
<tr>
<td>ETF_MODE</td>
<td>2803</td>
</tr>
<tr>
<td>ETF_PIDR0</td>
<td>2813</td>
</tr>
<tr>
<td>ETF_PIDR1</td>
<td>2814</td>
</tr>
<tr>
<td>ETF_PIDR2</td>
<td>2814</td>
</tr>
<tr>
<td>ETF_PIDR3</td>
<td>2815</td>
</tr>
<tr>
<td>ETF_PIDR4</td>
<td>2813</td>
</tr>
<tr>
<td>ETF_PSCR</td>
<td>2808</td>
</tr>
<tr>
<td>ETF_RRD</td>
<td>2800</td>
</tr>
<tr>
<td>ETF_RRP</td>
<td>2801</td>
</tr>
</tbody>
</table>
### Index

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETF_RSZ</td>
<td>2799</td>
</tr>
<tr>
<td>ETF_RWD</td>
<td>2803</td>
</tr>
<tr>
<td>ETF_RWP</td>
<td>2801</td>
</tr>
<tr>
<td>ETF_STS</td>
<td>2799</td>
</tr>
<tr>
<td>ETF_TRG</td>
<td>2802</td>
</tr>
<tr>
<td>EXTI_CPUEMR1</td>
<td>746</td>
</tr>
<tr>
<td>EXTI_CPUEMR2</td>
<td>748</td>
</tr>
<tr>
<td>EXTI_CPUEMR3</td>
<td>751</td>
</tr>
<tr>
<td>EXTI_CPUIMR1</td>
<td>746</td>
</tr>
<tr>
<td>EXTI_CPUIMR2</td>
<td>747</td>
</tr>
<tr>
<td>EXTI_CPUIMR3</td>
<td>749</td>
</tr>
<tr>
<td>EXTI_CPUIMR3</td>
<td>749</td>
</tr>
<tr>
<td>EXTI_CPUIMR3</td>
<td>752</td>
</tr>
<tr>
<td>EXTI_D3PCR2H</td>
<td>742</td>
</tr>
<tr>
<td>EXTI_FTSR1</td>
<td>735</td>
</tr>
<tr>
<td>EXTI_FTSR2</td>
<td>739</td>
</tr>
<tr>
<td>EXTI_FTSR3</td>
<td>743</td>
</tr>
<tr>
<td>EXTI_RTSR1</td>
<td>735</td>
</tr>
<tr>
<td>EXTI_RTSR2</td>
<td>738</td>
</tr>
<tr>
<td>EXTI_RTSR3</td>
<td>743</td>
</tr>
<tr>
<td>EXTI_SRDPMR3</td>
<td>745</td>
</tr>
<tr>
<td>EXTI_SRDPCR1H</td>
<td>738</td>
</tr>
<tr>
<td>EXTI_SRDPCR1L</td>
<td>737</td>
</tr>
<tr>
<td>EXTI_SRDPCR2L</td>
<td>741</td>
</tr>
<tr>
<td>EXTI_SRDPCR3H</td>
<td>745</td>
</tr>
<tr>
<td>EXTI_SRDPMR1</td>
<td>736</td>
</tr>
<tr>
<td>EXTI_SRDPMR2</td>
<td>741</td>
</tr>
<tr>
<td>EXTI_SWIER1</td>
<td>736</td>
</tr>
<tr>
<td>EXTI_SWIER2</td>
<td>740</td>
</tr>
<tr>
<td>EXTI_SWIER3</td>
<td>744</td>
</tr>
</tbody>
</table>

### F

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCCAN_CCU_CCFG</td>
<td>2523</td>
</tr>
<tr>
<td>FCCAN_CCU_CREL</td>
<td>2523</td>
</tr>
<tr>
<td>FCCAN_CCU_CSTAT</td>
<td>2525</td>
</tr>
<tr>
<td>FCCAN_CCU_CWD</td>
<td>2525</td>
</tr>
<tr>
<td>FCCAN_CCU_IE</td>
<td>2527</td>
</tr>
<tr>
<td>FCCAN_CCU_IR</td>
<td>2526</td>
</tr>
<tr>
<td>FCCAN_CCCR</td>
<td>2464</td>
</tr>
<tr>
<td>FDCAN_CREL</td>
<td>2461</td>
</tr>
<tr>
<td>FDCAN_DBTP</td>
<td>2461</td>
</tr>
<tr>
<td>FDCAN_ECR</td>
<td>2469</td>
</tr>
<tr>
<td>FDCAN_ENDN</td>
<td>2461</td>
</tr>
<tr>
<td>FDCAN_GFC</td>
<td>2480</td>
</tr>
<tr>
<td>FDCAN_HPMS</td>
<td>2483</td>
</tr>
<tr>
<td>FDCAN_IE</td>
<td>2476</td>
</tr>
<tr>
<td>FDCAN_IIE</td>
<td>2479</td>
</tr>
<tr>
<td>FDCAN_ILE</td>
<td>2478</td>
</tr>
<tr>
<td>FDCAN_IR</td>
<td>2473</td>
</tr>
<tr>
<td>FDCAN_NBTP</td>
<td>2466</td>
</tr>
<tr>
<td>Symbol</td>
<td>Page</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td>FDCAN_NDAT1</td>
<td>2483</td>
</tr>
<tr>
<td>FDCAN_NDAT2</td>
<td>2484</td>
</tr>
<tr>
<td>FDCAN_PSR</td>
<td>2470</td>
</tr>
<tr>
<td>FDCAN_RWD</td>
<td>2463</td>
</tr>
<tr>
<td>FDCAN_RXBC</td>
<td>2486</td>
</tr>
<tr>
<td>FDCAN_RXESC</td>
<td>2489</td>
</tr>
<tr>
<td>FDCAN_RXF0A</td>
<td>2486</td>
</tr>
<tr>
<td>FDCAN_RXF0C</td>
<td>2484</td>
</tr>
<tr>
<td>FDCAN_RXF0S</td>
<td>2485</td>
</tr>
<tr>
<td>FDCAN_RXF1A</td>
<td>2489</td>
</tr>
<tr>
<td>FDCAN_RXF1C</td>
<td>2487</td>
</tr>
<tr>
<td>FDCAN_RXF1S</td>
<td>2486</td>
</tr>
<tr>
<td>FDCAN_SIDFC</td>
<td>2481</td>
</tr>
<tr>
<td>FDCAN_TDCR</td>
<td>2472</td>
</tr>
<tr>
<td>FDCAN_TEST</td>
<td>2462</td>
</tr>
<tr>
<td>FDCAN_TOCC</td>
<td>2468</td>
</tr>
<tr>
<td>FDCAN_TOCV</td>
<td>2469</td>
</tr>
<tr>
<td>FDCAN_TSCC</td>
<td>2467</td>
</tr>
<tr>
<td>FDCAN_TSCV</td>
<td>2467</td>
</tr>
<tr>
<td>FDCAN_TTCPT</td>
<td>2516</td>
</tr>
<tr>
<td>FDCAN_TTCSM</td>
<td>2516</td>
</tr>
<tr>
<td>FDCAN_TTCTC</td>
<td>2515</td>
</tr>
<tr>
<td>FDCAN_TTGTP</td>
<td>2506</td>
</tr>
<tr>
<td>FDCAN_TTIE</td>
<td>2509</td>
</tr>
<tr>
<td>FDCAN_TTILS</td>
<td>2511</td>
</tr>
<tr>
<td>FDCAN_TTIR</td>
<td>2507</td>
</tr>
<tr>
<td>FDCAN_TTLGT</td>
<td>2515</td>
</tr>
<tr>
<td>FDCAN_TTMLM</td>
<td>2501</td>
</tr>
<tr>
<td>FDCAN_TTOCF</td>
<td>2500</td>
</tr>
<tr>
<td>FDCAN_TTOCN</td>
<td>2504</td>
</tr>
<tr>
<td>FDCAN_TTOST</td>
<td>2513</td>
</tr>
<tr>
<td>FDCAN_TTRMC</td>
<td>2499</td>
</tr>
<tr>
<td>FDCAN_TTTMC</td>
<td>2498</td>
</tr>
<tr>
<td>FDCAN_TTTMK</td>
<td>2506</td>
</tr>
<tr>
<td>FDCAN_TTTS</td>
<td>2517</td>
</tr>
<tr>
<td>FDCAN_TURCF</td>
<td>2502</td>
</tr>
<tr>
<td>FDCAN_TURNA</td>
<td>2514</td>
</tr>
<tr>
<td>FDCAN_TXBAR</td>
<td>2494</td>
</tr>
<tr>
<td>FDCAN_TXBC</td>
<td>2490</td>
</tr>
<tr>
<td>FDCAN_TXBCF</td>
<td>2495</td>
</tr>
<tr>
<td>FDCAN_TXBCIE</td>
<td>2496</td>
</tr>
<tr>
<td>FDCAN_TXBCR</td>
<td>2494</td>
</tr>
<tr>
<td>FDCAN_TXBRP</td>
<td>2493</td>
</tr>
<tr>
<td>FDCAN_TXBTIE</td>
<td>2495</td>
</tr>
<tr>
<td>FDCAN_TXBITO</td>
<td>2495</td>
</tr>
<tr>
<td>FDCAN_TXEFA</td>
<td>2498</td>
</tr>
<tr>
<td>FDCAN_TXEFC</td>
<td>2496</td>
</tr>
<tr>
<td>FDCAN_TXEFS</td>
<td>2497</td>
</tr>
<tr>
<td>FDCAN_TXESC</td>
<td>2492</td>
</tr>
<tr>
<td>FDCAN_TXFQS</td>
<td>2491</td>
</tr>
<tr>
<td>FDCAN_XIDAM</td>
<td>2482</td>
</tr>
<tr>
<td>FDCAN_XIDFC</td>
<td>2481</td>
</tr>
<tr>
<td>Symbol</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
</tr>
<tr>
<td>FLASH_ACR</td>
<td>203</td>
</tr>
<tr>
<td>FLASH_BOOT7_CURR</td>
<td>222</td>
</tr>
<tr>
<td>FLASH_BOOT7_PRGR</td>
<td>223</td>
</tr>
<tr>
<td>FLASH_CCR1</td>
<td>211</td>
</tr>
<tr>
<td>FLASH_CCR2</td>
<td>236</td>
</tr>
<tr>
<td>FLASH_CR1</td>
<td>204</td>
</tr>
<tr>
<td>FLASH_CR2</td>
<td>229</td>
</tr>
<tr>
<td>FLASH_CRCCECR1</td>
<td>224</td>
</tr>
<tr>
<td>FLASH_CRCCECR2</td>
<td>241</td>
</tr>
<tr>
<td>FLASH_CRCCEADD1R</td>
<td>226</td>
</tr>
<tr>
<td>FLASH_CRCCEADD2R</td>
<td>243</td>
</tr>
<tr>
<td>FLASH_CRCSCADD1R</td>
<td>225</td>
</tr>
<tr>
<td>FLASH_CRCSCADD2R</td>
<td>242</td>
</tr>
<tr>
<td>FLASH_ECC_FA1R</td>
<td>227</td>
</tr>
<tr>
<td>FLASH_ECC_FA2R</td>
<td>243</td>
</tr>
<tr>
<td>FLASH_KEYR1</td>
<td>203</td>
</tr>
<tr>
<td>FLASH_KEYR2</td>
<td>228</td>
</tr>
<tr>
<td>FLASH_OPTCCR</td>
<td>218</td>
</tr>
<tr>
<td>FLASH_OPTCR</td>
<td>212</td>
</tr>
<tr>
<td>FLASH_OPTKEYR</td>
<td>204</td>
</tr>
<tr>
<td>FLASH_OPTSRCUR</td>
<td>213</td>
</tr>
<tr>
<td>FLASH_OPTSR_PRG</td>
<td>216</td>
</tr>
<tr>
<td>FLASH_OTPBLCUR</td>
<td>227</td>
</tr>
<tr>
<td>FLASH_OTPBL_PRG</td>
<td>228</td>
</tr>
<tr>
<td>FLASH_PRAR_CUR1</td>
<td>219</td>
</tr>
<tr>
<td>FLASH_PRAR_CUR2</td>
<td>237</td>
</tr>
<tr>
<td>FLASH_PRAR_PRG1</td>
<td>219</td>
</tr>
<tr>
<td>FLASH_PRAR_PRG2</td>
<td>237</td>
</tr>
<tr>
<td>FLASH_SCAR_CUR1</td>
<td>220</td>
</tr>
<tr>
<td>FLASH_SCAR_CUR2</td>
<td>238</td>
</tr>
<tr>
<td>FLASH_SCAR_PRG1</td>
<td>221</td>
</tr>
<tr>
<td>FLASH_SCAR_PRG2</td>
<td>239</td>
</tr>
<tr>
<td>FLASH_SR1</td>
<td>208</td>
</tr>
<tr>
<td>FLASH_SR2</td>
<td>233</td>
</tr>
<tr>
<td>FLASH_WPSGN_CUR1R</td>
<td>221</td>
</tr>
<tr>
<td>FLASH_WPSGN_CUR2R</td>
<td>240</td>
</tr>
<tr>
<td>FLASH_WPSGN_PRG1R</td>
<td>222</td>
</tr>
<tr>
<td>FLASH_WPSGN_PRG2R</td>
<td>240</td>
</tr>
<tr>
<td>FMC_BCRx</td>
<td>821</td>
</tr>
<tr>
<td>FMC_BTRx</td>
<td>825</td>
</tr>
<tr>
<td>FMC_BWTRx</td>
<td>828</td>
</tr>
<tr>
<td>FMC_ECCR</td>
<td>841</td>
</tr>
<tr>
<td>FMC_PATT</td>
<td>840</td>
</tr>
<tr>
<td>FMC_PCR</td>
<td>836</td>
</tr>
<tr>
<td>FMC_PMEM</td>
<td>839</td>
</tr>
<tr>
<td>FMC_SDCMR</td>
<td>856</td>
</tr>
<tr>
<td>FMC_SDCRx</td>
<td>853</td>
</tr>
<tr>
<td>FMC_SDRTR</td>
<td>857</td>
</tr>
<tr>
<td>FMC_SDSR</td>
<td>859</td>
</tr>
<tr>
<td>FMC_SDTx</td>
<td>854</td>
</tr>
<tr>
<td>FMC_SR</td>
<td>837</td>
</tr>
</tbody>
</table>
## Index

### G

- GFXMMU CR ........................................ 765
- GPIOx_AFRH ..................................... 517
- GPIOx_AFRL ..................................... 516
- GPIOx_BSRR ..................................... 514
- GPIOx_IDR ....................................... 514
- GPIOx_LCKR ..................................... 515
- GPIOx_MODER .................................... 512
- GPIOx_ODR ....................................... 514
- GPIOx_OSCEDR ................................... 513
- GPIOx_OTYPER .................................... 512
- GPIOx_PUPDR ..................................... 513

### H

- HASH_CR ........................................... 1406
- HASH_CSR0 ....................................... 1414
- HASH_CSRx ....................................... 1414
- HASH_DIN ........................................ 1409
- HASH_HR0 ........................................ 1411
- HASH_HR5 ........................................ 1412
- HASH_HRx ........................................ 1411
- HASH_IMR ........................................ 1412
- HASH_SR .......................................... 1413
- HASH_STR ......................................... 1410
- HSEM_CR .......................................... 499
- HSEM_ICR ......................................... 498
- HSEM_IER ......................................... 498
- HSEM_ISR ......................................... 498
- HSEM_KEYR ....................................... 500
- HSEM_MISR ....................................... 499
- HSEM_RLRx ....................................... 497
- HSEM_Rx .......................................... 496

### I

- I2C_CR1 ............................................ 1934
- I2C_CR2 ............................................ 1937
- I2C_IER .......................................... 1946
- I2C_ISR .......................................... 1944
- I2C_OAR1 ......................................... 1940
- I2C_OAR2 ......................................... 1941
- I2C_PECR ......................................... 1947
- I2C_RXDR .......................................... 1948
- I2C_TIMEOUTR .................................... 1943
- I2C_TIMINGR ..................................... 1942
- I2C_TXDR .......................................... 1948-1949
- I2C_CR2 .......................................... 1937
- IWDG_KR .......................................... 1811
- IWDG_PR .......................................... 1812
- IWDG_RLR ......................................... 1813
- IWDG_SR .......................................... 1814
<table>
<thead>
<tr>
<th>Index</th>
<th>RM0455</th>
</tr>
</thead>
<tbody>
<tr>
<td>IWDG_WINR</td>
<td>1815</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>J</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>JPEG_CFR</td>
<td>1299</td>
</tr>
<tr>
<td>JPEG_CONFR0</td>
<td>1294</td>
</tr>
<tr>
<td>JPEG_CONFR1</td>
<td>1294</td>
</tr>
<tr>
<td>JPEG_CONFR2</td>
<td>1295</td>
</tr>
<tr>
<td>JPEG_CONFR3</td>
<td>1296</td>
</tr>
<tr>
<td>JPEG_CONFR4-7</td>
<td>1296</td>
</tr>
<tr>
<td>JPEG_CR</td>
<td>1297</td>
</tr>
<tr>
<td>JPEG_DIR</td>
<td>1300</td>
</tr>
<tr>
<td>JPEG_DOR</td>
<td>1300</td>
</tr>
<tr>
<td>JPEG_HUFFMIN</td>
<td>1301</td>
</tr>
<tr>
<td>JPEG_SR</td>
<td>1298</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM_ARR</td>
<td>1797</td>
</tr>
<tr>
<td>LPTIM_CFG</td>
<td>1792</td>
</tr>
<tr>
<td>LPTIM_CFG2</td>
<td>1798</td>
</tr>
<tr>
<td>LPTIM_CMP</td>
<td>1796</td>
</tr>
<tr>
<td>LPTIM_CNT</td>
<td>1797</td>
</tr>
<tr>
<td>LPTIM_CR</td>
<td>1795</td>
</tr>
<tr>
<td>LPTIM_ICR</td>
<td>1791</td>
</tr>
<tr>
<td>LPTIM_IER</td>
<td>1791</td>
</tr>
<tr>
<td>LPTIM_ISR</td>
<td>1790</td>
</tr>
<tr>
<td>LPUART_BRR</td>
<td>2078</td>
</tr>
<tr>
<td>LPUART_CR1</td>
<td>2067, 2070</td>
</tr>
<tr>
<td>LPUART_CR2</td>
<td>2073</td>
</tr>
<tr>
<td>LPUART_CR3</td>
<td>2075</td>
</tr>
<tr>
<td>LPUART_ICR</td>
<td>2087</td>
</tr>
<tr>
<td>LPUART_ISR</td>
<td>2079, 2084</td>
</tr>
<tr>
<td>LPUART_PARSER</td>
<td>2089</td>
</tr>
<tr>
<td>LPUART_RDR</td>
<td>2088</td>
</tr>
<tr>
<td>LPUART_RQR</td>
<td>2079</td>
</tr>
<tr>
<td>LPUART_TDR</td>
<td>2088</td>
</tr>
<tr>
<td>LTDC_AWC</td>
<td>1268</td>
</tr>
<tr>
<td>LTDC_BC</td>
<td>1271</td>
</tr>
<tr>
<td>LTDC_BPCR</td>
<td>1267</td>
</tr>
<tr>
<td>LTDC_CDSR</td>
<td>1275</td>
</tr>
<tr>
<td>LTDC_CPSR</td>
<td>1274</td>
</tr>
<tr>
<td>LTDC_GCR</td>
<td>1269</td>
</tr>
<tr>
<td>LTDC_ICR</td>
<td>1273</td>
</tr>
<tr>
<td>LTDC_IER</td>
<td>1272</td>
</tr>
<tr>
<td>LTDC_ISR</td>
<td>1273</td>
</tr>
<tr>
<td>LTDC_LIPCR</td>
<td>1274</td>
</tr>
<tr>
<td>LTDC_LxBFC</td>
<td>1281</td>
</tr>
<tr>
<td>LTDC_LxCACR</td>
<td>1279</td>
</tr>
<tr>
<td>LTDC_LxCFBAR</td>
<td>1282</td>
</tr>
<tr>
<td>LTDC_LxCFBLNR</td>
<td>1283</td>
</tr>
<tr>
<td>LTDC_LxCFBLR</td>
<td>1282</td>
</tr>
<tr>
<td>LTDC_LxCFCK</td>
<td>1278</td>
</tr>
</tbody>
</table>
LTDC_LxCLUTWR .......................... 1284
LTDC_LxCR .................................. 1275
LTDC_LxDCCCR ................................ 1280
LTDC_LxPFCCR ................................ 1278
LTDC_LxWHPCR ................................ 1276
LTDC_LxWVPCR ................................ 1277
LTDC_SRCR .................................. 1271
LTDC_SSCR .................................. 1267
LTDC_TWCR .................................. 1269

M

M7_CPUROM_CIDR0 .......................... 2863
M7_CPUROM_CIDR1 .......................... 2863
M7_CPUROM_CIDR2 .......................... 2864
M7_CPUROM_CIDR3 .......................... 2864
M7_CPUROM_MEMTYPE ......................... 2860
M7_CPUROM_PIDR0 .......................... 2861
M7_CPUROM_PIDR1 .......................... 2861
M7_CPUROM_PIDR2 .......................... 2862
M7_CPUROM_PIDR3 .......................... 2862
M7_CPUROM_PIDR4 .......................... 2860
M7_DWT_CIDR0 .............................. 2882
M7_DWT_CIDR1 .............................. 2882
M7_DWT_CIDR2 .............................. 2883
M7_DWT_CIDR3 .............................. 2883
M7_DWT_CMPx ................................. 2877
M7_DWT_CPICNT ............................. 2875
M7_DWT_CTRL ................................. 2873
M7_DWT_CYCCNT ............................. 2874
M7_DWT_EXCCNT ............................. 2875
M7_DWT_FOLDCNT ............................ 2876
M7_DWT_FUNCTx .............................. 2878
M7_DWT_LSUCNT .............................. 2876
M7_DWT_MASKx ................................. 2877
M7_DWT_PCSR ................................. 2877
M7_DWT_PIDR0 ............................... 2880
M7_DWT_PIDR1 ............................... 2880
M7_DWT_PIDR2 ............................... 2881
M7_DWT_PIDR3 ............................... 2881
M7_DWT_PIDR4 ............................... 2879
M7_DWT_SLPCNT .............................. 2875
M7_ETM_AUTHTSTAT ........................ 2924
M7_ETM_CCCTL ............................... 2908
M7_ETM_CIDR0 ............................... 2929
M7_ETM_CIDR1 ............................... 2929
M7_ETM_CIDR2 ............................... 2930
M7_ETM_CIDR3 ............................... 2930
M7_ETM_CLAIMCLR .......................... 2923
M7_ETM_CLAIMSET ........................... 2922
M7_ETM_CNTRLDV ............................ 2911
M7_ETM_CONFIG .............................. 2904
M7_ETM_DEVARCH ................. 2925
M7_ETM_DEVTYPE ................. 2926
M7_ETM_EVENTCTL0 ............... 2905
M7_ETM_EVENTCTL1 ............... 2906
M7_ETM_IDR0 ..................... 2914
M7_ETM_IDR1 ..................... 2915
M7_ETM_IDR10 .................... 2912
M7_ETM_IDR11 .................... 2912
M7_ETM_IDR12 .................... 2913
M7_ETM_IDR13 .................... 2913
M7_ETM_IDR2 ..................... 2915
M7_ETM_IDR3 ..................... 2916
M7_ETM_IDR4 ..................... 2917
M7_ETM_IDR5 ..................... 2918
M7_ETM_IDR8 ..................... 2911
M7_ETM_IDR9 ..................... 2912
M7_ETM_IMSPEC0 .................. 2913
M7_ETM_LAR ...................... 2923
M7_ETM_LSR ...................... 2924
M7_ETM_PDC ...................... 2921
M7_ETM_PDS ...................... 2922
M7_ETM_PIDR0 .................... 2927
M7_ETM_PIDR1 .................... 2927
M7_ETM_PIDR2 .................... 2928
M7_ETM_PIDR3 .................... 2928
M7_ETM_PIDR4 .................... 2926
M7_ETM_PRGCTL ................... 2903
M7_ETM_RSCTL2 ................... 2919
M7_ETM_RSCTL3 ................... 2919
M7_ETM_SSCC0 .................... 2920
M7_ETM_SSCS0 .................... 2920
M7_ETM_SSPCIC0 .................. 2921
M7_ETMSTALLCCTL ................. 2907
M7_ETM_STAT ..................... 2904
M7_ETM_SYNCP .................... 2907
M7_ETM_TRACEID .................. 2908
M7_ETM_VICTL .................... 2909
M7_ETM_VIPCSSCTL ............... 2910
M7_ETM_VISSCTL ................. 2910
M7_FPB_CIDR0 .................... 2900
M7_FPB_CIDR1 .................... 2900
M7_FPB_CIDR2 .................... 2901
M7_FPB_CIDR3 .................... 2901
M7_FPB_COMPx ................... 2896
M7_FPB_CTRL .................... 2895
M7_FPB_PIDR0 .................... 2898
M7_FPB_PIDR1 .................... 2898
M7_FPB_PIDR2 .................... 2899
M7_FPB_PIDR3 .................... 2899
M7_FPB_PIDR4 .................... 2897
M7_FPB_REMAP ................... 2896
M7_ITM_CIDR0 .................... 2892
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>M7_ITM_CIDR1</td>
<td>2892</td>
</tr>
<tr>
<td>M7_ITM_CIDR2</td>
<td>2893</td>
</tr>
<tr>
<td>M7_ITM_CIDR3</td>
<td>2893</td>
</tr>
<tr>
<td>M7_ITM_PIDR0</td>
<td>2890</td>
</tr>
<tr>
<td>M7_ITM_PIDR1</td>
<td>2890</td>
</tr>
<tr>
<td>M7_ITM_PIDR2</td>
<td>2891</td>
</tr>
<tr>
<td>M7_ITM_PIDR3</td>
<td>2891</td>
</tr>
<tr>
<td>M7_ITM_PIDR4</td>
<td>2889</td>
</tr>
<tr>
<td>M7_ITM_STIMx</td>
<td>2887</td>
</tr>
<tr>
<td>M7_ITM_TCR</td>
<td>2888</td>
</tr>
<tr>
<td>M7_ITM_TER</td>
<td>2887</td>
</tr>
<tr>
<td>M7_ITM_TPR</td>
<td>2888</td>
</tr>
<tr>
<td>M7_PPBROM_CIDR0</td>
<td>2869</td>
</tr>
<tr>
<td>M7_PPBROM_CIDR1</td>
<td>2869</td>
</tr>
<tr>
<td>M7_PPBROM_CIDR2</td>
<td>2870</td>
</tr>
<tr>
<td>M7_PPBROM_CIDR3</td>
<td>2870</td>
</tr>
<tr>
<td>M7_PPBROM_MEMTYPE</td>
<td>2866</td>
</tr>
<tr>
<td>M7_PPBROM_PIDR0</td>
<td>2867</td>
</tr>
<tr>
<td>M7_PPBROM_PIDR1</td>
<td>2867</td>
</tr>
<tr>
<td>M7_PPBROM_PIDR2</td>
<td>2868</td>
</tr>
<tr>
<td>M7_PPBROM_PIDR3</td>
<td>2868</td>
</tr>
<tr>
<td>M7_PPBROM_PIDR4</td>
<td>2866</td>
</tr>
<tr>
<td>MDOIOS_CLRFR</td>
<td>2312</td>
</tr>
<tr>
<td>MDOIOS_CR</td>
<td>2308</td>
</tr>
<tr>
<td>MDOIOS_CCR</td>
<td>2310</td>
</tr>
<tr>
<td>MDOIOS_CWRFR</td>
<td>2309</td>
</tr>
<tr>
<td>MDOIOS_DINRx</td>
<td>2313</td>
</tr>
<tr>
<td>MDOIOS_DOUTRx</td>
<td>2313</td>
</tr>
<tr>
<td>MDOIOS_RDFR</td>
<td>2310</td>
</tr>
<tr>
<td>MDOIOS_SR</td>
<td>2311</td>
</tr>
<tr>
<td>MDOIOS_WRFR</td>
<td>2309</td>
</tr>
<tr>
<td>MDMA_CxBNDTR</td>
<td>582</td>
</tr>
<tr>
<td>MDMA_CxBRUR</td>
<td>586</td>
</tr>
<tr>
<td>MDMA_CxCR</td>
<td>577</td>
</tr>
<tr>
<td>MDMA_CxDAR</td>
<td>584</td>
</tr>
<tr>
<td>MDMA_CxESR</td>
<td>575</td>
</tr>
<tr>
<td>MDMA_CxFCSR</td>
<td>574</td>
</tr>
<tr>
<td>MDMA_CxISR</td>
<td>573</td>
</tr>
<tr>
<td>MDMA_CxLAR</td>
<td>587</td>
</tr>
<tr>
<td>MDMA_CxMAR</td>
<td>589</td>
</tr>
<tr>
<td>MDMA_CxMDR</td>
<td>589</td>
</tr>
<tr>
<td>MDMA_CxSAR</td>
<td>584</td>
</tr>
<tr>
<td>MDMA_CxTBR</td>
<td>588</td>
</tr>
<tr>
<td>MDMA_CxTCR</td>
<td>579</td>
</tr>
<tr>
<td>MDMA_GISR0</td>
<td>572</td>
</tr>
<tr>
<td>OCTOSPI_PSMKR</td>
<td>899</td>
</tr>
<tr>
<td>OCTOSPI_ABR</td>
<td>904</td>
</tr>
<tr>
<td>OCTOSPI_AR</td>
<td>898</td>
</tr>
<tr>
<td>OCTOSPI_CCR</td>
<td>900</td>
</tr>
<tr>
<td>OCTOSPI_CR</td>
<td>889</td>
</tr>
<tr>
<td>--------------------</td>
<td>------</td>
</tr>
<tr>
<td>OCTOSPI_DCR1</td>
<td>892</td>
</tr>
<tr>
<td>OCTOSPI_DCR2</td>
<td>893</td>
</tr>
<tr>
<td>OCTOSPI_DCR3</td>
<td>894</td>
</tr>
<tr>
<td>OCTOSPI_DCR4</td>
<td>895</td>
</tr>
<tr>
<td>OCTOSPI_DLR</td>
<td>897</td>
</tr>
<tr>
<td>OCTOSPI_DR</td>
<td>898</td>
</tr>
<tr>
<td>OCTOSPI_FCR</td>
<td>897</td>
</tr>
<tr>
<td>OCTOSPI_HLCR</td>
<td>912</td>
</tr>
<tr>
<td>OCTOSPI_LPTR</td>
<td>903</td>
</tr>
<tr>
<td>OCTOSPI_PIR</td>
<td>900</td>
</tr>
<tr>
<td>OCTOSPI_PSMAR</td>
<td>900</td>
</tr>
<tr>
<td>OCTOSPI_SR</td>
<td>896</td>
</tr>
<tr>
<td>OCTOSPI_TCR</td>
<td>903</td>
</tr>
<tr>
<td>OCTOSPI_WABR</td>
<td>912</td>
</tr>
<tr>
<td>OCTOSPI_WCCR</td>
<td>909</td>
</tr>
<tr>
<td>OCTOSPI_WIR</td>
<td>912</td>
</tr>
<tr>
<td>OCTOSPI_WPABR</td>
<td>908</td>
</tr>
<tr>
<td>OCTOSPI_WPCCR</td>
<td>905</td>
</tr>
<tr>
<td>OCTOSPI_WPIR</td>
<td>908</td>
</tr>
<tr>
<td>OCTOSPI_WPTCR</td>
<td>907</td>
</tr>
<tr>
<td>OCTOSPI_WTCR</td>
<td>911</td>
</tr>
<tr>
<td>OCTOSPIIOM_CR</td>
<td>920</td>
</tr>
<tr>
<td>OCTOSPIIOM_PnCR</td>
<td>920</td>
</tr>
<tr>
<td>OCTOSPIIOM_PnCR</td>
<td>920</td>
</tr>
<tr>
<td>OPAMP_COR</td>
<td>1144</td>
</tr>
<tr>
<td>OPAMP1_CSR</td>
<td>1141</td>
</tr>
<tr>
<td>OPAMP1_HSOTR</td>
<td>1144</td>
</tr>
<tr>
<td>OPAMP1_OTR</td>
<td>1143</td>
</tr>
<tr>
<td>OPAMP2_CSR</td>
<td>1144</td>
</tr>
<tr>
<td>OPAMP2_HSOTR</td>
<td>1147</td>
</tr>
<tr>
<td>OPAMP2_OTR</td>
<td>1146</td>
</tr>
<tr>
<td>OTFDEC_ICR</td>
<td>1433</td>
</tr>
<tr>
<td>OTFDEC_IER</td>
<td>1433</td>
</tr>
<tr>
<td>OTFDEC_ISR</td>
<td>1431</td>
</tr>
<tr>
<td>OTFDEC_RxFCGR</td>
<td>1427</td>
</tr>
<tr>
<td>OTFDEC_RxENDADDR</td>
<td>1428</td>
</tr>
<tr>
<td>OTFDEC_RxKEYR0</td>
<td>1430</td>
</tr>
<tr>
<td>OTFDEC_RxKEYR1</td>
<td>1430</td>
</tr>
<tr>
<td>OTFDEC_RxKEYR2</td>
<td>1430</td>
</tr>
<tr>
<td>OTFDEC_RxKEYR3</td>
<td>1431</td>
</tr>
<tr>
<td>OTFDEC_RxNONCER0</td>
<td>1429</td>
</tr>
<tr>
<td>OTFDEC_RxNONCER1</td>
<td>1429</td>
</tr>
<tr>
<td>OTFDEC2_RxSTARTADDR</td>
<td>1428</td>
</tr>
<tr>
<td>OTG_CID</td>
<td>2586</td>
</tr>
<tr>
<td>OTG_DAINT</td>
<td>2616</td>
</tr>
<tr>
<td>OTG_DAINTMSK</td>
<td>2617</td>
</tr>
<tr>
<td>OTG_DCFG</td>
<td>2609</td>
</tr>
<tr>
<td>OTG_DCTL</td>
<td>2611</td>
</tr>
<tr>
<td>OTG_DEAHINT</td>
<td>2620</td>
</tr>
</tbody>
</table>
OTG_DEACHINTMSK .............. 2621
OTG_DIEPCTLx .................. 2624
OTG_DIEPDMAx .................. 2628
OTG_DIEPEMPMSK ............... 2620
OTG_DIEPINTx .................. 2626
OTG_DIEPMSK .................. 2614
OTG_DIEPTSIZ0 ................. 2628
OTG_DIEPTSIZx ................. 2629
OTG_DIEPTXF0 ................. 2582
OTG_DIEPTXFx ................. 2590
OTG_DOEPCTL0 ................. 2630
OTG_DOEPCTLx ................. 2635
OTG_DOEPDMAx ................. 2635
OTG_DOEPINTx ................. 2632
OTG_DOEPMSK ................. 2615
OTG_DOEPSTSIZ0 ............... 2634
OTG_DOEPSTSIZx ............... 2638
OTG_DSTS ..................... 2613
OTG_DTHRCtl .................. 2619
OTG_DTXFSSTx ................. 2629
OTG_DVBUSDIS ................. 2618
OTG_DVBUSPULSE ............... 2618
OTG_GAHBCFG .................. 2563
OTG_GCCFG ................... 2584
OTG_GINTMSK .................. 2574
OTG_GINTSTS ................. 2570
OTG_GLPMCFG .................. 2586
OTG_GOTGCTL ................. 2558
OTG_GOTGINT .................. 2561
OTG_GRSTCTL ................. 2568
OTG_GRXFSIZ ................. 2582
OTG_GRXSTSP ............... 2580-2581
OTG_GRXSTSR .............. 2578-2579
OTG_GUSBCFG .................. 2564
OTG_HAINT ................... 2595
OTG_HAINTMSK .................. 2595
OTG_HCCHARx .................. 2599
OTG_HCDMABx ................. 2608
OTG_HCDMASGx ................. 2607
OTG_HCDMAx ................. 2607
OTG_HCFG .................. 2591
OTG_HCINTMSK ................. 2602
OTG_HCINTx .................. 2601
OTG_HCSPLTx ................. 2600
OTG_HCTSIZSGx ................. 2605
OTG_HCTSIZx ................. 2604
OTG_HFIR ................ 2592
OTG_HFLBADDR ................. 2596
OTG_HFNUM .................. 2593
OTG_HNPTXFSIZ ................. 2582
OTG_HNPTXSTS ................. 2583
OTG_HPRT ............... 2596
Index

OTG_HPTXFSIZ .......................... 2590
OTG_HPTXSTS .......................... 2594
OTG_HS_DIEPEACHMSK1 ................. 2621
OTG_HS_DOEPEACHMSK1 .................. 2622
OTG_PGCCTL ............................ 2639

P

PSSI_CR ................................. 1246
PSSI_DR ................................. 1251
PSSI_ICR ................................. 1250
PSSI_IER ................................. 1249
PSSI_MIS ................................. 1249
PSSI_RIS ................................. 1248
PSSI_SR ................................. 1247
PWR_CPUCCR ............................. 312
PWR_CR1 .................................. 306
PWR_CR2 .................................. 309
PWR_CR3 .................................. 310
PWR_CSR1 .................................. 308
PWR_SRDCR ............................... 313
PWR_WKUPCR .............................. 314
PWR_WKUPEPR ............................ 315
PWR_WKUPFR .............................. 315

R

RAMECC_IER ............................. 140
RAMECC_MxFCR ........................... 141
RAMECC_MxFAR ........................... 142
RAMECC_MxFDRH .......................... 143
RAMECC_MxFDRL .......................... 142
RAMECC_MxFECR .......................... 143
RAMECC_MxSR ............................ 141
RCC_AHB1ENR ............................ 440
RCC_AHB1LPENR ........................... 457
RCC_AHB1RSTR ........................... 419
RCC_AHB2ENR ............................ 442
RCC_AHB2LPENR ........................... 458
RCC_AHB2RSTR ........................... 420
RCC_AHB3ENR ............................ 439
RCC_AHB3LPENR ........................... 454
RCC_AHB3RSTR ........................... 417
RCC_AHB4ENR ............................ 443
RCC_AHB4LPENR ........................... 459
RCC_AHB4RSTR ........................... 421
RCC_APB1HENR ........................... 449
RCC_APB1LPENR ........................... 465
RCC_APB1HRSTR ........................... 426
RCC_APB1LENR ........................... 446
RCC_APB1LLPENR ........................... 462
RCC_APB1LRSTR ........................... 423
RCC_APB2ENR ............................ 450
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCC_APB2LPENR</td>
<td></td>
<td>466</td>
</tr>
<tr>
<td>RCC_APB2RSTR</td>
<td></td>
<td>427</td>
</tr>
<tr>
<td>RCC_APB3ENR</td>
<td></td>
<td>445</td>
</tr>
<tr>
<td>RCC_APB3LPENR</td>
<td></td>
<td>461</td>
</tr>
<tr>
<td>RCC_APB3RSTR</td>
<td></td>
<td>423</td>
</tr>
<tr>
<td>RCC_APB4ENR</td>
<td></td>
<td>452</td>
</tr>
<tr>
<td>RCC_APB4LPENR</td>
<td></td>
<td>468</td>
</tr>
<tr>
<td>RCC_APB4RSTR</td>
<td></td>
<td>429</td>
</tr>
<tr>
<td>RCC_BDCR</td>
<td></td>
<td>415</td>
</tr>
<tr>
<td>RCC_CDCR1</td>
<td></td>
<td>408</td>
</tr>
<tr>
<td>RCC_CDCCIP2R</td>
<td></td>
<td>406</td>
</tr>
<tr>
<td>RCC_CDCCIPR</td>
<td></td>
<td>402</td>
</tr>
<tr>
<td>RCC_CDCFG1R</td>
<td></td>
<td>389</td>
</tr>
<tr>
<td>RCC_CDCFG2R</td>
<td></td>
<td>391</td>
</tr>
<tr>
<td>RCC_CFGR</td>
<td></td>
<td>387</td>
</tr>
<tr>
<td>RCC_CICR</td>
<td></td>
<td>413</td>
</tr>
<tr>
<td>RCC_CIER</td>
<td></td>
<td>410</td>
</tr>
<tr>
<td>RCC_CIFR</td>
<td></td>
<td>411</td>
</tr>
<tr>
<td>RCC_CKGAENR</td>
<td></td>
<td>434</td>
</tr>
<tr>
<td>RCC_CR</td>
<td></td>
<td>381</td>
</tr>
<tr>
<td>RCC_CRRCR</td>
<td></td>
<td>385</td>
</tr>
<tr>
<td>RCC_CSSCFG1R</td>
<td></td>
<td>386</td>
</tr>
<tr>
<td>RCC_CSR</td>
<td></td>
<td>417</td>
</tr>
<tr>
<td>RCC_HSICFG1R</td>
<td></td>
<td>385</td>
</tr>
<tr>
<td>RCC_PLL1DIVR</td>
<td></td>
<td>396</td>
</tr>
<tr>
<td>RCC_PLL1FRACR</td>
<td></td>
<td>398</td>
</tr>
<tr>
<td>RCC_PLL2DIVR</td>
<td></td>
<td>398</td>
</tr>
<tr>
<td>RCC_PLL2FRACR</td>
<td></td>
<td>400</td>
</tr>
<tr>
<td>RCC_PLL2DIVR</td>
<td></td>
<td>400</td>
</tr>
<tr>
<td>RCC_PLL3DIVR</td>
<td></td>
<td>402</td>
</tr>
<tr>
<td>RCC_PLL3FRACR</td>
<td></td>
<td>394</td>
</tr>
<tr>
<td>RCC_PLLCFGR</td>
<td></td>
<td>392</td>
</tr>
<tr>
<td>RCC_PLLCKSEL</td>
<td></td>
<td>437</td>
</tr>
<tr>
<td>RCC_RSR</td>
<td></td>
<td>437</td>
</tr>
<tr>
<td>RCC_SRDMR</td>
<td></td>
<td>431</td>
</tr>
<tr>
<td>RCC_SRDCCIP1R</td>
<td></td>
<td>408</td>
</tr>
<tr>
<td>RCC_SRDCCIP2R</td>
<td></td>
<td>391</td>
</tr>
<tr>
<td>RNG_C</td>
<td></td>
<td>1319</td>
</tr>
<tr>
<td>RNG_DR</td>
<td></td>
<td>1322</td>
</tr>
<tr>
<td>RNG_HTCR</td>
<td></td>
<td>1322</td>
</tr>
<tr>
<td>RNG_SR</td>
<td></td>
<td>1321</td>
</tr>
<tr>
<td>RTC_ALRMAR</td>
<td></td>
<td>1849</td>
</tr>
<tr>
<td>RTC_ALRMASSR</td>
<td></td>
<td>1850</td>
</tr>
<tr>
<td>RTC_ALRMBR</td>
<td></td>
<td>1851</td>
</tr>
<tr>
<td>RTC_ALRMBSSR</td>
<td></td>
<td>1852</td>
</tr>
<tr>
<td>RTC_CALR</td>
<td></td>
<td>1845</td>
</tr>
<tr>
<td>RTC_CFG1R</td>
<td></td>
<td>1855</td>
</tr>
<tr>
<td>RTC_CFG2R</td>
<td></td>
<td>1841</td>
</tr>
<tr>
<td>RTC_DR</td>
<td></td>
<td>1837</td>
</tr>
<tr>
<td>RTC_ICSR</td>
<td></td>
<td>1838</td>
</tr>
<tr>
<td>RTC_MISR</td>
<td></td>
<td>1853</td>
</tr>
<tr>
<td>RTC_PRER</td>
<td></td>
<td>1840</td>
</tr>
<tr>
<td>RTC_SCR</td>
<td></td>
<td>1854</td>
</tr>
<tr>
<td>Symbol</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>RTC_SHIFTR</td>
<td>1846</td>
<td></td>
</tr>
<tr>
<td>RTC_SR</td>
<td>1852</td>
<td></td>
</tr>
<tr>
<td>RTC_SSR</td>
<td>1838</td>
<td></td>
</tr>
<tr>
<td>RTC_TR</td>
<td>1835</td>
<td></td>
</tr>
<tr>
<td>RTC_TSDR</td>
<td>1847</td>
<td></td>
</tr>
<tr>
<td>RTC_TSSSR</td>
<td>1848</td>
<td></td>
</tr>
<tr>
<td>RTC_TSTR</td>
<td>1847</td>
<td></td>
</tr>
<tr>
<td>RTC_WPR</td>
<td>1844</td>
<td></td>
</tr>
<tr>
<td>RTC_WUTR</td>
<td>1840</td>
<td></td>
</tr>
<tr>
<td>SAI_ACLRFR</td>
<td>2227</td>
<td></td>
</tr>
<tr>
<td>SAI_ACR1</td>
<td>2206</td>
<td></td>
</tr>
<tr>
<td>SAI_ACR2</td>
<td>2212</td>
<td></td>
</tr>
<tr>
<td>SAI_ADR</td>
<td>2229</td>
<td></td>
</tr>
<tr>
<td>SAI_AFRCR</td>
<td>2216</td>
<td></td>
</tr>
<tr>
<td>SAI_AIM</td>
<td>2220</td>
<td></td>
</tr>
<tr>
<td>SAI_ASLOTR</td>
<td>2218</td>
<td></td>
</tr>
<tr>
<td>SAI_ASR</td>
<td>2223</td>
<td></td>
</tr>
<tr>
<td>SAI_BCLRFR</td>
<td>2228</td>
<td></td>
</tr>
<tr>
<td>SAI_BCR1</td>
<td>2209</td>
<td></td>
</tr>
<tr>
<td>SAI_BCR2</td>
<td>2214</td>
<td></td>
</tr>
<tr>
<td>SAI_BDR</td>
<td>2230</td>
<td></td>
</tr>
<tr>
<td>SAI_BFRCR</td>
<td>2217</td>
<td></td>
</tr>
<tr>
<td>SAI_BIM</td>
<td>2222</td>
<td></td>
</tr>
<tr>
<td>SAI_BSLOTR</td>
<td>2219</td>
<td></td>
</tr>
<tr>
<td>SAI_BSR</td>
<td>2225</td>
<td></td>
</tr>
<tr>
<td>SAI_GCR</td>
<td>2206</td>
<td></td>
</tr>
<tr>
<td>SAI_PDMCR</td>
<td>2230</td>
<td></td>
</tr>
<tr>
<td>SAI_PDMDLY</td>
<td>2231</td>
<td></td>
</tr>
<tr>
<td>SDMMC_ACKTIMER</td>
<td>2390</td>
<td></td>
</tr>
<tr>
<td>SDMMC_ARGR</td>
<td>2375</td>
<td></td>
</tr>
<tr>
<td>SDMMC_CLKCR</td>
<td>2373</td>
<td></td>
</tr>
<tr>
<td>SDMMC_CMDR</td>
<td>2375</td>
<td></td>
</tr>
<tr>
<td>SDMMC_DCNTR</td>
<td>2381</td>
<td></td>
</tr>
<tr>
<td>SDMMC_DCTRL</td>
<td>2380</td>
<td></td>
</tr>
<tr>
<td>SDMMC_DLENR</td>
<td>2379</td>
<td></td>
</tr>
<tr>
<td>SDMMC_DTIMER</td>
<td>2378</td>
<td></td>
</tr>
<tr>
<td>SDMMC_FIFORx</td>
<td>2390</td>
<td></td>
</tr>
<tr>
<td>SDMMC_ICR</td>
<td>2385</td>
<td></td>
</tr>
<tr>
<td>SDMMC_IDMABASE0R</td>
<td>2392</td>
<td></td>
</tr>
<tr>
<td>SDMMC_IDMABASE1R</td>
<td>2393</td>
<td></td>
</tr>
<tr>
<td>SDMMC_IDMABSIZER</td>
<td>2392</td>
<td></td>
</tr>
<tr>
<td>SDMMC_IDMACTRLR</td>
<td>2391</td>
<td></td>
</tr>
<tr>
<td>SDMMC_MASKR</td>
<td>2387</td>
<td></td>
</tr>
<tr>
<td>SDMMC_POWER</td>
<td>2372</td>
<td></td>
</tr>
<tr>
<td>SDMMC_RESPCMDR</td>
<td>2377</td>
<td></td>
</tr>
<tr>
<td>SDMMC_RESPxR</td>
<td>2378</td>
<td></td>
</tr>
<tr>
<td>SMPMI_IER</td>
<td>2296</td>
<td></td>
</tr>
<tr>
<td>SPDIFRX_CR</td>
<td>2260</td>
<td></td>
</tr>
</tbody>
</table>
RM0455 Index

SPDIFRX_CSR .......................... 2270
SPDIFRX_DIR ................................ 2270
SPDIFRX_FMT0_DR .......................... 2267
SPDIFRX_FMT1_DR .................................. 2268
SPDIFRX_FMT2_DR .......................... 2269
SPDIFRX_IFCR ................................ 2266
SPDIFRX_IMR ................................ 2263
SPDIFRX_SR ................................ 2264
SPI_CFG1 ................................ 2150
SPI_CFG2 ................................ 2153
SPI_CR2 ................................ 2149
SPI_CRCPOLY .......................... 2160
SPI_I2SCFGR ................................ 2162
SPI_RXCRC ................................ 2161
SPI_TXCRC ................................ 2161
SPI_UDRDR ................................ 2162
SPI2S_CR1 ................................ 2148
SPI2S_IER ................................ 2155
SPI2S_IFCR ................................ 2158
SPI2S_RXDR ................................ 2160
SPI2S_SR ................................ 2156
SPI2S_TXDR ................................ 2159
SWO_AUTHSTAT .......................... 2842
SWO_CIDR0 ................................ 2847
SWO_CIDR1 ................................ 2847
SWO_CIDR2 ................................ 2848
SWO_CIDR3 ................................ 2848
SWO_CLAIMCLR ................................ 2841
SWO_CLAIMSET .......................... 2840
SWO_CODR ................................ 2839
SWO_DEVID ................................ 2843
SWO_DEVTYPE .......................... 2844
SWO_FFSR ................................ 2840
SWO_LAR ................................ 2841
SWO_LSR ................................ 2842
SWO_PIDR0 ................................ 2845
SWO_PIDR1 ................................ 2845
SWO_PIDR2 ................................ 2846
SWO_PIDR3 ................................ 2846
SWO_PIDR4 ................................ 2844
SWO_SPPR ................................ 2839
SWPMI_BRR ................................ 2293
SWPMI_CR ................................ 2292
SWPMI_ICR ................................ 2295
SWPMI_ISR ................................ 2294
SWPMI_OR ................................ 2299
SWPMI_RDR ................................ 2298
SWPMI_RFL ................................ 2298
SWPMI_TDR ................................ 2298
SYSCFG_CCCR .................................. 532
SYSCFG_CCCSR .................................. 530
SYSCFG_CCVR .................................. 531
Index

SYSCFG_CFGR ............................... 528
SYSCFG_EXTICR1 ............................ 525
SYSCFG_EXTICR2 ............................ 526
SYSCFG_EXTICR3 ............................ 527
SYSCFG_EXTICR4 ............................ 528
SYSCFG_PMCR ............................... 523
SYSROM_CIDR0 ............................... 2763
SYSROM_CIDR1 ............................... 2763
SYSROM_CIDR2 ............................... 2763
SYSROM_CIDR3 ............................... 2764
SYSROM_MEMTYPE ........................... 2760
SYSROM_PIDR0 ............................... 2761
SYSROM_PIDR1 ............................... 2761
SYSROM_PIDR2 ............................... 2762
SYSROM_PIDR3 ............................... 2762
SYSROM_PIDR4 ............................... 2760

T
TAMP_ATCR1 ............................... 1869
TAMP_ATOR .................................. 1872
TAMP_ATSEEDR ............................... 1871
TAMP_BKPxR ................................. 1878
TAMP_CFGR .................................. 1877
TAMP_COUNTR ................................. 1877
TAMP_CR1 .................................... 1866
TAMP_CR2 .................................... 1867
TAMP_FLTCR ................................. 1868
TAMP_IER .................................... 1872
TAMP_MISR .................................. 1875
TAMP_SCR .................................... 1876
TAMP_SR ..................................... 1874
TIM1_AF1 ................................. 1531
TIM1_AF2 ................................. 1533
TIM1_TISEL .................................. 1538
TIM12_ARR ................................. 1657
TIM12_CCER ................................. 1655
TIM12_CCMR1 ............................... 1651-1652
TIM12_CCR1 ................................. 1657
TIM12_CCR2 ................................. 1658
TIM12_CNT ................................. 1656
TIM12_CR1 ................................. 1645
TIM12_CR2 ................................. 1646
TIM12_DIER ................................. 1649
TIM12_EGR ................................. 1650
TIM12_PSC ................................. 1657
TIM12_SMCR ................................. 1647
TIM12_SR ................................. 1649
TIM12_TISEL ............................... 1658
TIM13_TISEL ............................... 1669
TIM14_TISEL ............................... 1669
TIM15_AFI ................................. 1733
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM15_ARR</td>
<td>1727</td>
</tr>
<tr>
<td>TIM15_BDTR</td>
<td>1729</td>
</tr>
<tr>
<td>TIM15_CCER</td>
<td>1724</td>
</tr>
<tr>
<td>TIM15_CCMR1</td>
<td>1720-1721</td>
</tr>
<tr>
<td>TIM15_CCR1</td>
<td>1728</td>
</tr>
<tr>
<td>TIM15_CCR2</td>
<td>1729</td>
</tr>
<tr>
<td>TIM15_CNT</td>
<td>1727</td>
</tr>
<tr>
<td>TIM15_CR1</td>
<td>1712</td>
</tr>
<tr>
<td>TIM15_CR2</td>
<td>1713</td>
</tr>
<tr>
<td>TIM15_DCR</td>
<td>1732</td>
</tr>
<tr>
<td>TIM15_DIER</td>
<td>1716</td>
</tr>
<tr>
<td>TIM15_DMAR</td>
<td>1732</td>
</tr>
<tr>
<td>TIM15_EGR</td>
<td>1719</td>
</tr>
<tr>
<td>TIM15_PSC</td>
<td>1727</td>
</tr>
<tr>
<td>TIM15_RCR</td>
<td>1728</td>
</tr>
<tr>
<td>TIM15_SMCR</td>
<td>1715</td>
</tr>
<tr>
<td>TIM15_SR</td>
<td>1717</td>
</tr>
<tr>
<td>TIM15_TISEL</td>
<td>1734</td>
</tr>
<tr>
<td>TIM16_AF1</td>
<td>1755</td>
</tr>
<tr>
<td>TIM16_TISEL</td>
<td>1756</td>
</tr>
<tr>
<td>TIM17_AF1</td>
<td>1757</td>
</tr>
<tr>
<td>TIM17_TISEL</td>
<td>1758</td>
</tr>
<tr>
<td>TIM2_AF1</td>
<td>1612</td>
</tr>
<tr>
<td>TIM2_TISEL</td>
<td>1613</td>
</tr>
<tr>
<td>TIM3_AF1</td>
<td>1612</td>
</tr>
<tr>
<td>TIM3_TISEL</td>
<td>1614</td>
</tr>
<tr>
<td>TIM4_AF1</td>
<td>1612</td>
</tr>
<tr>
<td>TIM4_TISEL</td>
<td>1615</td>
</tr>
<tr>
<td>TIM5_AF1</td>
<td>1613</td>
</tr>
<tr>
<td>TIM5_TISEL</td>
<td>1616</td>
</tr>
<tr>
<td>TIM8_AF1</td>
<td>1534</td>
</tr>
<tr>
<td>TIM8_AF2</td>
<td>1536</td>
</tr>
<tr>
<td>TIM8_TISEL</td>
<td>1538</td>
</tr>
<tr>
<td>TIMx_ARR</td>
<td>1520, 1608, 1668, 1749, 1772</td>
</tr>
<tr>
<td>TIMx_BDTR</td>
<td>1523, 1751</td>
</tr>
<tr>
<td>TIMx_CCER</td>
<td>1517, 1605, 1667, 1746</td>
</tr>
<tr>
<td>TIMx_CCMR1</td>
<td>1510-1511, 1599, 1601, 1664-1665, 1743-1744</td>
</tr>
<tr>
<td>TIMx_CCMR2</td>
<td>1514-1515, 1603-1604</td>
</tr>
<tr>
<td>TIMx_CCMR3</td>
<td>1529</td>
</tr>
<tr>
<td>TIMx_CCR1</td>
<td>1521, 1608, 1669, 1750</td>
</tr>
<tr>
<td>TIMx_CCR2</td>
<td>1522, 1609</td>
</tr>
<tr>
<td>TIMx_CCR3</td>
<td>1522, 1609</td>
</tr>
<tr>
<td>TIMx_CCR4</td>
<td>1523, 1610</td>
</tr>
<tr>
<td>TIMx_CCR5</td>
<td>1530</td>
</tr>
<tr>
<td>TIMx_CCR6</td>
<td>1531</td>
</tr>
<tr>
<td>TIMx_CNT</td>
<td>1520, 1606-1607, 1668, 1748, 1771</td>
</tr>
<tr>
<td>TIMx_CR1</td>
<td>1499, 1589, 1661, 1738, 1768</td>
</tr>
<tr>
<td>TIMx_CR2</td>
<td>1500, 1590, 1739, 1770</td>
</tr>
<tr>
<td>TIMx_DCR</td>
<td>1527, 1611, 1754</td>
</tr>
<tr>
<td>TIMx_DIER</td>
<td>1505, 1595, 1662, 1740, 1770</td>
</tr>
<tr>
<td>TIMx_DMAR</td>
<td>1528, 1611, 1754</td>
</tr>
<tr>
<td>TIMx_EGR</td>
<td>1509, 1597, 1663, 1742, 1771</td>
</tr>
<tr>
<td>Index</td>
<td>RM0455</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>TIMx_PSC</td>
<td>1520, 1607, 1668, 1749, 1772</td>
</tr>
<tr>
<td>TIMx_RCR</td>
<td>1521, 1750</td>
</tr>
<tr>
<td>TIMx_SMC R</td>
<td>1503, 1592</td>
</tr>
<tr>
<td>TIMx_SR</td>
<td>1507, 1596, 1662, 1741, 1771</td>
</tr>
<tr>
<td>TPIU_AUTHSTAT</td>
<td>2829</td>
</tr>
<tr>
<td>TPIU_CIDR0</td>
<td>2834</td>
</tr>
<tr>
<td>TPIU_CIDR1</td>
<td>2834</td>
</tr>
<tr>
<td>TPIU_CIDR2</td>
<td>2835</td>
</tr>
<tr>
<td>TPIU_CIDR3</td>
<td>2835</td>
</tr>
<tr>
<td>TPIU_CLAIMCLR</td>
<td>2828</td>
</tr>
<tr>
<td>TPIU_CLAIMSET</td>
<td>2827</td>
</tr>
<tr>
<td>TPIU_CURPSIZE</td>
<td>2820</td>
</tr>
<tr>
<td>TPIU_CURTPM</td>
<td>2824</td>
</tr>
<tr>
<td>TPIU_DEVID</td>
<td>2830</td>
</tr>
<tr>
<td>TPIU_DEVTYPE</td>
<td>2831</td>
</tr>
<tr>
<td>TPIU_FFCR</td>
<td>2826</td>
</tr>
<tr>
<td>TPIU_FFSR</td>
<td>2825</td>
</tr>
<tr>
<td>TPIU_FSCR</td>
<td>2827</td>
</tr>
<tr>
<td>TPIU_LAR</td>
<td>2828</td>
</tr>
<tr>
<td>TPIU_LSR</td>
<td>2829</td>
</tr>
<tr>
<td>TPIU_PIDR0</td>
<td>2832</td>
</tr>
<tr>
<td>TPIU_PIDR1</td>
<td>2832</td>
</tr>
<tr>
<td>TPIU_PIDR2</td>
<td>2833</td>
</tr>
<tr>
<td>TPIU_PIDR3</td>
<td>2833</td>
</tr>
<tr>
<td>TPIU_PIDR4</td>
<td>2831</td>
</tr>
<tr>
<td>TPIU_SUPPSIZE</td>
<td>2820</td>
</tr>
<tr>
<td>TPIU_SUPTPM</td>
<td>2823</td>
</tr>
<tr>
<td>TPIU_SUPTRGM</td>
<td>2821</td>
</tr>
<tr>
<td>TPIU_TPRCR</td>
<td>2825</td>
</tr>
<tr>
<td>TPIU_TRGCNT</td>
<td>2822</td>
</tr>
<tr>
<td>TPIU_TRGMULT</td>
<td>2822</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>USART_BRR</td>
</tr>
<tr>
<td>USART_CR1</td>
</tr>
<tr>
<td>USART_CR2</td>
</tr>
<tr>
<td>USART_CR3</td>
</tr>
<tr>
<td>USART_GTPR</td>
</tr>
<tr>
<td>USART_ICR</td>
</tr>
<tr>
<td>USART_ISR</td>
</tr>
<tr>
<td>USART_PRES C</td>
</tr>
<tr>
<td>USART_RDR</td>
</tr>
<tr>
<td>USART_RQR</td>
</tr>
<tr>
<td>USART_RTOR</td>
</tr>
<tr>
<td>USART_TDR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREFBUF_CCR</td>
</tr>
<tr>
<td>VREFBUF_CSR</td>
</tr>
</tbody>
</table>
W

WWDG_CFR ...........................1805
WWDG_CR ...........................1804
WWDG_SR ...........................1806

XYZ

DMA2D_BGCLUT .....................709
DMA2D_FGCLUT .....................708
JPEG_DHTMEM ........................1303
JPEG_HUFFBASE .....................1302
JPEG_HUFFENC_ACx ..................1304
JPEG_HUFFENC_DCx ..................1304
JPEG_HUFFSYM .......................1303
JPEG_QMEMx ........................1301
IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, please refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2020 STMicroelectronics – All rights reserved