Introduction

This reference manual complements the datasheets of the STM32G0x1 microcontrollers, providing information required for application and in particular for software development. It pertains to the superset of feature sets available on STM32G0x1 microcontrollers.

For feature set, ordering information, and mechanical and electrical characteristics of a particular STM32G0x1 device, refer to its corresponding datasheet.

For information on the Arm® Cortex®-M0+ core, refer to the Cortex®-M0+ technical reference manual.

Related documents

- PM0223 programming manual for Cortex®-M0+ core(a)
- STM32G0x1 datasheets(a)
- AN2606 application note on booting STM32 MCUs(a)

---

a. Available on STMicroelectronics website www.st.com
Contents

1 Documentation conventions ........................................ 50
   1.1 General information ........................................ 50
   1.2 List of abbreviations for registers ............................. 50
   1.3 Glossary ................................................... 51
   1.4 Availability of peripherals .................................... 51

2 Memory and bus architecture ..................................... 52
   2.1 System architecture .......................................... 52
   2.2 Memory organization ......................................... 54
      2.2.1 Introduction ............................................ 54
      2.2.2 Memory map and register boundary addresses .......... 55
   2.3 Embedded SRAM .............................................. 59
   2.4 Flash memory overview ....................................... 60
   2.5 Boot configuration .......................................... 60

3 Embedded Flash memory (FLASH) ................................. 62
   3.1 FLASH Introduction .......................................... 62
   3.2 FLASH main features ......................................... 62
   3.3 FLASH functional description ................................ 62
      3.3.1 FLASH memory organization ......................... 62
      3.3.2 FLASH empty check ..................................... 64
      3.3.3 FLASH error code correction (ECC) ............... 64
      3.3.4 FLASH read access latency............................ 64
      3.3.5 FLASH memory acceleration ........................... 65
      3.3.6 FLASH program and erase operations .................. 66
      3.3.7 FLASH Main memory erase sequences .................. 67
      3.3.8 FLASH Main memory programming sequences ........... 68
   3.4 FLASH option bytes .......................................... 72
      3.4.1 FLASH option byte description ....................... 72
      3.4.2 FLASH option byte programming ....................... 78
   3.5 FLASH memory protection .................................... 79
      3.5.1 FLASH read protection (RDP) .......................... 79
      3.5.2 FLASH proprietary code readout protection (PCROP) .... 82
3.5.3 Flash write protection (WRP) ........................................... 84
3.5.4 Securable memory area ................................................. 85
3.5.5 Disabling core debug access ............................................ 86
3.5.6 Forcing boot from Flash memory ...................................... 86

3.6 Flash interrupts ............................................................. 86

3.7 Flash registers ............................................................... 88
3.7.1 Flash access control register (FLASH_ACR) ...................... 88
3.7.2 Flash key register (FLASH_KEYR) ...................................... 89
3.7.3 Flash option key register (FLASH_OPTKEYR) .................... 89
3.7.4 Flash status register (FLASH_SR) ....................................... 90
3.7.5 Flash control register (FLASH_CR) ..................................... 91
3.7.6 Flash ECC register (FLASH_ECCR) .................................... 93
3.7.7 Flash option register (FLASH_OPTR) ................................... 94
3.7.8 Flash PCROP area A start address register (FLASH_PCROP1ASR) ........................................... 96
3.7.9 Flash PCROP area A end address register (FLASH_PCROP1AER) ........................................... 96
3.7.10 Flash WRP area A address register (FLASH_WRP1AR) ......... 97
3.7.11 Flash WRP area B address register (FLASH_WRP1BR) ......... 98
3.7.12 Flash PCROP area B start address register (FLASH_PCROP1BSR) ........................................... 98
3.7.13 Flash PCROP area B end address register (FLASH_PCROP1BER) ........................................... 99
3.7.14 Flash security register (FLASH_SECR) .............................. 99
3.7.15 Flash register map ..................................................... 100

4 Power control (PWR) ......................................................... 102
4.1 Power supplies ............................................................ 102
4.1.1 ADC and DAC reference voltage ..................................... 103
4.1.2 Battery backup of RTC domain ....................................... 103
4.1.3 Voltage regulator ....................................................... 105
4.1.4 Dynamic voltage scaling management .............................. 106

4.2 Power supply supervisor .................................................. 107
4.2.1 Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR) ........................................... 107
4.2.2 Programmable voltage detector (PVD) ............................ 108

4.3 Low-power modes ......................................................... 109
4.3.1 Run mode ................................................................. 114
4.3.2 Low-power run mode (LP run) ........................................ 114
4.3.3 Low-power modes .................................................. 115
4.3.4 Sleep mode ....................................................... 116
4.3.5 Low-power sleep mode (LP sleep) ............................ 117
4.3.6 Stop 0 mode ...................................................... 118
4.3.7 Stop 1 mode ...................................................... 121
4.3.8 Standby mode .................................................... 122
4.3.9 Shutdown mode ................................................... 124
4.3.10 Auto-wake up from low-power mode ........................ 125
4.4 PWR registers .......................................................... 126
  4.4.1 Power control register 1 (PWR_CR1) ......................... 126
  4.4.2 Power control register 2 (PWR_CR2) ......................... 127
  4.4.3 Power control register 3 (PWR_CR3) ......................... 129
  4.4.4 Power control register 4 (PWR_CR4) ......................... 130
  4.4.5 Power status register 1 (PWR_SR1) ......................... 131
  4.4.6 Power status register 2 (PWR_SR2) ......................... 132
  4.4.7 Power status clear register (PWR_SCR) .................... 133
  4.4.8 Power Port A pull-up control register (PWR_PUCRA) ...... 134
  4.4.9 Power Port A pull-down control register (PWR_PDCRA) .... 134
  4.4.10 Power Port B pull-up control register (PWR_PUCRB) ...... 135
  4.4.11 Power Port B pull-down control register (PWR_PDCRB) .... 135
  4.4.12 Power Port C pull-up control register (PWR_PUCRC) ...... 136
  4.4.13 Power Port C pull-down control register (PWR_PDCRC) .... 136
  4.4.14 Power Port D pull-up control register (PWR_PUCRD) ...... 137
  4.4.15 Power Port D pull-down control register (PWR_PDCRD) .... 137
  4.4.16 Power Port F pull-up control register (PWR_PUCRF) ...... 138
  4.4.17 Power Port F pull-down control register (PWR_PDCRF) .... 138
  4.4.18 PWR register map and reset value table .................. 140

5 Reset and clock control (RCC) ....................................... 142
  5.1 Reset ............................................................. 142
    5.1.1 Power reset ................................................ 142
    5.1.2 System reset ............................................... 142
    5.1.3 RTC domain reset ........................................ 144
  5.2 Clocks ........................................................... 144
    5.2.1 HSE clock .................................................. 148
    5.2.2 HSI16 clock ................................................. 149
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.3</td>
<td>PLL</td>
<td>150</td>
</tr>
<tr>
<td>5.2.4</td>
<td>LSE clock</td>
<td>150</td>
</tr>
<tr>
<td>5.2.5</td>
<td>LSI clock</td>
<td>151</td>
</tr>
<tr>
<td>5.2.6</td>
<td>System clock (SYSCLK) selection</td>
<td>151</td>
</tr>
<tr>
<td>5.2.7</td>
<td>Clock source frequency versus voltage scaling</td>
<td>151</td>
</tr>
<tr>
<td>5.2.8</td>
<td>Clock security system (CSS)</td>
<td>152</td>
</tr>
<tr>
<td>5.2.9</td>
<td>Clock security system for LSE clock (LSECSS)</td>
<td>152</td>
</tr>
<tr>
<td>5.2.10</td>
<td>ADC clock</td>
<td>153</td>
</tr>
<tr>
<td>5.2.11</td>
<td>RTC clock</td>
<td>153</td>
</tr>
<tr>
<td>5.2.12</td>
<td>Timer clock</td>
<td>153</td>
</tr>
<tr>
<td>5.2.13</td>
<td>Watchdog clock</td>
<td>154</td>
</tr>
<tr>
<td>5.2.14</td>
<td>Clock-out capability</td>
<td>154</td>
</tr>
<tr>
<td>5.2.15</td>
<td>Internal/external clock measurement with TIM14/TIM16/TIM17</td>
<td>154</td>
</tr>
<tr>
<td>5.2.16</td>
<td>Peripheral clock enable registers</td>
<td>157</td>
</tr>
<tr>
<td>5.3</td>
<td>Low-power modes</td>
<td>157</td>
</tr>
<tr>
<td>5.4</td>
<td>RCC registers</td>
<td>158</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Clock control register (RCC_CR)</td>
<td>158</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Internal clock sources calibration register (RCC_ICSCR)</td>
<td>160</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Clock configuration register (RCC_CFGR)</td>
<td>160</td>
</tr>
<tr>
<td>5.4.4</td>
<td>PLL configuration register (RCC_PLLCFGR)</td>
<td>162</td>
</tr>
<tr>
<td>5.4.5</td>
<td>Clock interrupt enable register (RCC_CIER)</td>
<td>165</td>
</tr>
<tr>
<td>5.4.6</td>
<td>Clock interrupt flag register (RCC_CIFR)</td>
<td>166</td>
</tr>
<tr>
<td>5.4.7</td>
<td>Clock interrupt clear register (RCC_CICR)</td>
<td>167</td>
</tr>
<tr>
<td>5.4.8</td>
<td>I/O port reset register (RCC_IOPRSTR)</td>
<td>168</td>
</tr>
<tr>
<td>5.4.9</td>
<td>AHB peripheral reset register (RCC_AHBRSTR)</td>
<td>169</td>
</tr>
<tr>
<td>5.4.10</td>
<td>APB peripheral reset register 1 (RCC_APBSTR1)</td>
<td>170</td>
</tr>
<tr>
<td>5.4.11</td>
<td>APB peripheral reset register 2 (RCC_APBSTR2)</td>
<td>173</td>
</tr>
<tr>
<td>5.4.12</td>
<td>I/O port clock enable register (RCC_IOPENR)</td>
<td>174</td>
</tr>
<tr>
<td>5.4.13</td>
<td>AHB peripheral clock enable register (RCC_AHBENR)</td>
<td>175</td>
</tr>
<tr>
<td>5.4.14</td>
<td>APB peripheral clock enable register 1 (RCC_APBENR1)</td>
<td>176</td>
</tr>
<tr>
<td>5.4.15</td>
<td>APB peripheral clock enable register 2 (RCC_APBENR2)</td>
<td>179</td>
</tr>
<tr>
<td>5.4.16</td>
<td>I/O port in Sleep mode clock enable register (RCC_IOPSMENR)</td>
<td>180</td>
</tr>
<tr>
<td>5.4.17</td>
<td>AHB peripheral clock enable in Sleep/Stop mode register (RCC_AHBSMENR)</td>
<td>181</td>
</tr>
<tr>
<td>5.4.18</td>
<td>APB peripheral clock enable in Sleep/Stop mode register 1 (RCC_APBSMENR1)</td>
<td>182</td>
</tr>
<tr>
<td>5.4.19</td>
<td>APB peripheral clock enable in Sleep/Stop mode register 2 (RCC_APBSMENR2)</td>
<td>185</td>
</tr>
</tbody>
</table>
6 General-purpose I/Os (GPIO) ........................................ 196

6.1 Introduction ......................................................... 196
6.2 GPIO main features .................................................. 196
6.3 GPIO functional description ........................................ 196
  6.3.1 General-purpose I/O (GPIO) .................................. 198
  6.3.2 I/O pin alternate function multiplexer and mapping .............. 198
  6.3.3 I/O port control registers ..................................... 199
  6.3.4 I/O port data registers ........................................ 199
  6.3.5 I/O data bitwise handling ..................................... 200
  6.3.6 GPIO locking mechanism ....................................... 200
  6.3.7 I/O alternate function input/output ............................ 200
  6.3.8 External interrupt/wakeup lines ................................ 201
  6.3.9 Input configuration ............................................ 201
  6.3.10 Output configuration .......................................... 202
  6.3.11 Alternate function configuration ............................. 202
  6.3.12 Analog configuration .......................................... 203
  6.3.13 Using the HSE or LSE oscillator pins as GPIOs .................. 204
  6.3.14 Using the GPIO pins in the RTC domain ....................... 204
  6.3.15 USB PD / Dead battery support ................................ 204

6.4 GPIO registers ....................................................... 205
  6.4.1 GPIO port mode register (GPIOx_MODER) (x = A to D, F) .......... 205
  6.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to D, F) .. 205
  6.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to D, F) . 206
  6.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to D, F) . 206
  6.4.5 GPIO port input data register (GPIOx_IDR) (x = A to D, F) .......... 207
  6.4.6 GPIO port output data register (GPIOx_ODR) (x = A to D, F) ........ 207
  6.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to D, F) .... 207
  6.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to D, F) . 208
  6.4.9 GPIO alternate function low register (GPIOx_AFR[0..3]) (x = A to D, F) . 209
6.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to D, F) ........................................... 210
6.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to D, F) ............... 210
6.4.12 GPIO register map .................................................. 211

7 System configuration controller (SYSCFG) .......................... 213
7.1 SYSCFG registers ...................................................... 213
7.1.1 SYSCFG configuration register 1 (SYSCFG_CFRG1) ............... 213
7.1.2 SYSCFG configuration register 2 (SYSCFG_CFRG2) ............... 216
7.1.3 SYSCFG interrupt line 0 status register (SYSCFG_ITLINE0) ...... 218
7.1.4 SYSCFG interrupt line 1 status register (SYSCFG_ITLINE1) ...... 218
7.1.5 SYSCFG interrupt line 2 status register (SYSCFG_ITLINE2) ...... 219
7.1.6 SYSCFG interrupt line 3 status register (SYSCFG_ITLINE3) ...... 219
7.1.7 SYSCFG interrupt line 4 status register (SYSCFG_ITLINE4) ...... 219
7.1.8 SYSCFG interrupt line 5 status register (SYSCFG_ITLINE5) ...... 221
7.1.9 SYSCFG interrupt line 6 status register (SYSCFG_ITLINE6) ...... 221
7.1.10 SYSCFG interrupt line 7 status register (SYSCFG_ITLINE7) ...... 221
7.1.11 SYSCFG interrupt line 8 status register (SYSCFG_ITLINE8) ...... 222
7.1.12 SYSCFG interrupt line 9 status register (SYSCFG_ITLINE9) ...... 222
7.1.13 SYSCFG interrupt line 10 status register (SYSCFG_ITLINE10) ... 223
7.1.14 SYSCFG interrupt line 11 status register (SYSCFG_ITLINE11) ... 223
7.1.15 SYSCFG interrupt line 12 status register (SYSCFG_ITLINE12) ... 224
7.1.16 SYSCFG interrupt line 13 status register (SYSCFG_ITLINE13) ... 224
7.1.17 SYSCFG interrupt line 14 status register (SYSCFG_ITLINE14) ... 225
7.1.18 SYSCFG interrupt line 15 status register (SYSCFG_ITLINE15) ... 225
7.1.19 SYSCFG interrupt line 16 status register (SYSCFG_ITLINE16) ... 225
7.1.20 SYSCFG interrupt line 17 status register (SYSCFG_ITLINE17) ... 226
7.1.21 SYSCFG interrupt line 18 status register (SYSCFG_ITLINE18) ... 226
7.1.22 SYSCFG interrupt line 19 status register (SYSCFG_ITLINE19) ... 226
7.1.23 SYSCFG interrupt line 20 status register (SYSCFG_ITLINE20) ... 227
7.1.24 SYSCFG interrupt line 21 status register (SYSCFG_ITLINE21) ... 227
7.1.25 SYSCFG interrupt line 22 status register (SYSCFG_ITLINE22) ... 227
7.1.26 SYSCFG interrupt line 23 status register (SYSCFG_ITLINE23) ... 228
7.1.27 SYSCFG interrupt line 24 status register (SYSCFG_ITLINE24) ... 228
7.1.28 SYSCFG interrupt line 25 status register (SYSCFG_ITLINE25) ... 228
7.1.29 SYSCFG interrupt line 26 status register (SYSCFG_ITLINE26) ... 229
7.1.30 SYSCFG interrupt line 27 status register (SYSCFG_ITLINE27) ... 229
8 Interconnect matrix ........................................... 234
  8.1 Introduction ............................................. 234
  8.2 Connection summary ...................................... 234
  8.3 Interconnection details .................................. 235
    8.3.1 From TIM1, TIM2, TIM3, TIM15, TIM16, and TIM17, to TIM1, TIM2, TIM3, and TIM15 ............... 235
    8.3.2 From TIM1, TIM2, TIM3, TIM6, TIM15, and EXTI, to ADC ........................................ 236
    8.3.3 From ADC to TIM1 ................................... 236
    8.3.4 From TIM1, TIM2, TIM3, TIM6, TIM7, TIM15, LPTIM1, LPTIM2, and EXTI, to DAC .................. 237
    8.3.5 From HSE, LSE, LSI, MCO, RTC and TAMP, to TIM2, TIM14, TIM16, and TIM17 ....................... 237
    8.3.6 From RTC, TAMP, COMP1, and COMP2, to LPTIM1 and LPTIM2 .................................. 237
    8.3.7 From TIM1, TIM2, TIM3, and TIM15, to COMP1 and COMP2 ........................................ 238
    8.3.8 From internal analog sources to ADC ............... 238
    8.3.9 From COMP1 and COMP2, to TIM1, TIM2, TIM3, TIM15, TIM16, and TIM17 ....................... 239
    8.3.10 From system errors to TIM1, TIM2, TIM3, TIM15, TIM16, and TIM17 ....................... 239
    8.3.11 From TIM16, TIM17, USART1, and USART4, to IRTIM ........................................ 239
    8.3.12 From TIM14, LPTIM1, and LPTIM2, to DMAMUX ............................................... 240

9 Direct memory access controller (DMA) ....................... 241
  9.1 Introduction ............................................. 241
  9.2 DMA main features ...................................... 241
  9.3 DMA implementation .................................... 242
    9.3.1 DMA .................................................. 242
    9.3.2 DMA request mapping ................................ 242
  9.4 DMA functional description .............................. 242
    9.4.1 DMA block diagram ................................ 242
    9.4.2 DMA pins and internal signals ...................... 243
    9.4.3 DMA transfers ........................................ 243
    9.4.4 DMA arbitration ..................................... 244
## DMA request multiplexer (DMAMUX) .................................. 262

### 10.1 Introduction .................................................. 262

### 10.2 DMAMUX main features ........................................ 263

### 10.3 DMAMUX implementation ....................................... 263

#### 10.3.1 DMAMUX instantiation ..................................... 263

#### 10.3.2 DMAMUX mapping ........................................... 263

### 10.4 DMAMUX functional description ............................... 266

#### 10.4.1 DMAMUX block diagram .................................... 266

#### 10.4.2 DMAMUX signals ............................................ 267

#### 10.4.3 DMAMUX channels .......................................... 267

#### 10.4.4 DMAMUX request line multiplexer ......................... 267

#### 10.4.5 DMAMUX request generator ................................. 270

### 10.5 DMAMUX interrupts ............................................ 271

### 10.6 DMAMUX registers ............................................. 272

#### 10.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR) ........................................ 272

#### 10.6.2 DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR) ........................................ 273

#### 10.6.3 DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR) ........................................ 273

#### 10.6.4 DMAMUX request generator channel x configuration register (DMAMUX_RGxCR) ........................................ 274

#### 10.6.5 DMAMUX request generator interrupt status register (DMAMUX_RGSR) ........................................ 275
10.6.6 DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR) ........................................ 275
10.6.7 DMAMUX register map ........................................ 276

11 Nested vectored interrupt controller (NVIC) ....................... 278
11.1 Main features ........................................ 278
11.2 SysTick calibration value register ................................ 278
11.3 Interrupt and exception vectors ................................ 278

12 Extended interrupt and event controller (EXTI) .................... 281
12.1 EXTI main features ........................................ 281
12.2 EXTI block diagram ........................................ 281
  12.2.1 EXTI connections between peripherals and CPU ............... 283
12.3 EXTI functional description .................................... 283
  12.3.1 EXTI configurable event input wakeup ......................... 284
  12.3.2 EXTI direct event input wakeup ................................ 285
  12.3.3 EXTI mux ........................................ 285
12.4 EXTI functional behavior ...................................... 287
12.5 EXTI registers ........................................ 287
  12.5.1 EXTI rising trigger selection register (EXTI_RTSR1) ........ 288
  12.5.2 EXTI falling trigger selection register (EXTI_FTSR1) .......... 288
  12.5.3 EXTI software interrupt event register (EXTI_SWIER1) ....... 289
  12.5.4 EXTI rising edge pending register (EXTI_RPR1) .............. 289
  12.5.5 EXTI falling edge pending register (EXTI_FPR1) .............. 290
  12.5.6 EXTI external interrupt selection register (EXTI_EXTICRx) ... 290
  12.5.7 EXTI CPU wakeup with interrupt mask register (EXTI_IMR1) ... 291
  12.5.8 EXTI CPU wakeup with event mask register (EXTI_EMR1) ...... 292
  12.5.9 EXTI CPU wakeup with interrupt mask register (EXTI_IMR2) ... 293
  12.5.10 EXTI CPU wakeup with event mask register (EXTI_EMR2) ...... 294
  12.5.11 EXTI register map ...................................... 294

13 Cyclic redundancy check calculation unit (CRC) .................... 296
13.1 Introduction ........................................ 296
13.2 CRC main features ........................................ 296
13.3 CRC functional description ................................... 297
  13.3.1 CRC block diagram ...................................... 297
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.3.2</td>
<td>CRC internal signals</td>
</tr>
<tr>
<td>13.3.3</td>
<td>CRC operation</td>
</tr>
<tr>
<td>13.4</td>
<td>CRC registers</td>
</tr>
<tr>
<td>13.4.1</td>
<td>CRC data register (CRC_DR)</td>
</tr>
<tr>
<td>13.4.2</td>
<td>CRC independent data register (CRC_IDR)</td>
</tr>
<tr>
<td>13.4.3</td>
<td>CRC control register (CRC_CR)</td>
</tr>
<tr>
<td>13.4.4</td>
<td>CRC initial value (CRC_INIT)</td>
</tr>
<tr>
<td>13.4.5</td>
<td>CRC polynomial (CRC_POL)</td>
</tr>
<tr>
<td>13.4.6</td>
<td>CRC register map</td>
</tr>
<tr>
<td>14</td>
<td>Analog-to-digital converter (ADC)</td>
</tr>
<tr>
<td>14.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>14.2</td>
<td>ADC main features</td>
</tr>
<tr>
<td>14.3</td>
<td>ADC functional description</td>
</tr>
<tr>
<td>14.3.1</td>
<td>ADC pins and internal signals</td>
</tr>
<tr>
<td>14.3.2</td>
<td>ADC voltage regulator (ADVREGEN)</td>
</tr>
<tr>
<td>14.3.3</td>
<td>Calibration (ADCAL)</td>
</tr>
<tr>
<td>14.3.4</td>
<td>ADC on-off control (ADEN, ADDIS, ADRDY)</td>
</tr>
<tr>
<td>14.3.5</td>
<td>ADC clock (CKMODE, PRESC[3:0])</td>
</tr>
<tr>
<td>14.3.6</td>
<td>ADC connectivity</td>
</tr>
<tr>
<td>14.3.7</td>
<td>Configuring the ADC</td>
</tr>
<tr>
<td>14.3.8</td>
<td>Channel selection (CHSEL, SCANDIR, CHSELRMOD)</td>
</tr>
<tr>
<td>14.3.9</td>
<td>Programmable sampling time (SMPx[2:0])</td>
</tr>
<tr>
<td>14.3.10</td>
<td>Single conversion mode (CONT=0)</td>
</tr>
<tr>
<td>14.3.11</td>
<td>Continuous conversion mode (CONT=1)</td>
</tr>
<tr>
<td>14.3.12</td>
<td>Starting conversions (ADSTART)</td>
</tr>
<tr>
<td>14.3.13</td>
<td>Timings</td>
</tr>
<tr>
<td>14.3.14</td>
<td>Stopping an ongoing conversion (ADSTP)</td>
</tr>
<tr>
<td>14.4</td>
<td>Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)</td>
</tr>
<tr>
<td>14.4.1</td>
<td>Discontinuous mode (DISCEN)</td>
</tr>
<tr>
<td>14.4.2</td>
<td>Programmable resolution (RES) - fast conversion mode</td>
</tr>
<tr>
<td>14.4.3</td>
<td>End of conversion, end of sampling phase (EOC, EOSMP flags)</td>
</tr>
<tr>
<td>14.4.4</td>
<td>End of conversion sequence (EOS flag)</td>
</tr>
<tr>
<td>14.4.5</td>
<td>Example timing diagrams (single/continuous modes)</td>
</tr>
<tr>
<td>14.4.6</td>
<td>Low frequency trigger mode</td>
</tr>
<tr>
<td>14.5</td>
<td>Data management</td>
</tr>
</tbody>
</table>
14.5.1 Data register and data alignment (ADC_DR, ALIGN) .................. 323
14.5.2 ADC overrun (OVR, OVRMOD) .......................................... 323
14.5.3 Managing a sequence of data converted without using the DMA ...... 324
14.5.4 Managing converted data without using the DMA without overrun ... 324
14.5.5 Managing converted data using the DMA ............................... 324

14.6 Low-power features ............................................................. 326
14.6.1 Wait mode conversion ....................................................... 326
14.6.2 Auto-off mode (AUTOFF) .................................................. 327

14.7 Analog window watchdog (AWD1EN, AWD1SGL, AWD1CH,
ADC_AWDxCR, ADC_AWDxTR) .................................................. 328
14.7.1 Description of analog watchdog 1 ........................................ 328
14.7.2 Description of analog watchdog 2 and 3 ............................... 329
14.7.3 ADC_AWDx_OUT signal output generation .............................. 330
14.7.4 Analog Watchdog threshold control ...................................... 331

14.8 Oversampler ................................................................. 332
14.8.1 ADC operating modes supported when oversampling ................. 334
14.8.2 Analog watchdog ........................................................... 334
14.8.3 Triggered mode ............................................................. 334

14.9 Temperature sensor and internal reference voltage ......................... 335

14.10 Battery voltage monitoring .................................................. 337

14.11 ADC interrupts .............................................................. 338

14.12 ADC registers ............................................................... 340
14.12.1 ADC interrupt and status register (ADC_ISR) ....................... 340
14.12.2 ADC interrupt enable register (ADC_IER) ............................. 342
14.12.3 ADC control register (ADC_CR) ......................................... 344
14.12.4 ADC configuration register 1 (ADC_CFRG1) .......................... 346
14.12.5 ADC configuration register 2 (ADC_CFRG2) .......................... 350
14.12.6 ADC sampling time register (ADC_SMPr) .............................. 351
14.12.7 ADC watchdog threshold register (ADC_AWD1TR) .................. 352
14.12.8 ADC watchdog threshold register (ADC_AWD2TR) .................. 353
14.12.9 ADC channel selection register [alternate] (ADC_CHSelR) .......... 354
14.12.10 ADC channel selection register [alternate] (ADC_CHSelR) ......... 355
14.12.11 ADC watchdog threshold register (ADC_AWD3TR) ................. 356
14.12.12 ADC data register (ADC_DR) ........................................... 357
14.12.13 ADC Analog Watchdog 2 Configuration register (ADC_AWD2CR) .. 358
14.12.14 ADC Analog Watchdog 3 Configuration register (ADC_AWD3CR) .. 358
14.12.15 ADC Calibration factor (ADC_CALFACT) .................................................. 359
14.12.16 ADC common configuration register (ADC_CCR) ................................. 359
14.12.17 ADC register map .................................................................................. 361

15 Digital-to-analog converter (DAC) ................................................................. 363

15.1 Introduction ................................................................................................. 363
15.2 DAC main features .................................................................................... 363
15.3 DAC implementation .................................................................................. 364
15.4 DAC functional description ........................................................................ 365
  15.4.1 DAC block diagram .............................................................................. 365
  15.4.2 DAC pins and internal signals ............................................................... 366
  15.4.3 DAC channel enable ............................................................................ 367
  15.4.4 DAC data format .................................................................................. 367
  15.4.5 DAC conversion ................................................................................... 368
  15.4.6 DAC output voltage ............................................................................. 368
  15.4.7 DAC trigger selection .......................................................................... 369
  15.4.8 DMA requests ...................................................................................... 370
  15.4.9 Noise generation .................................................................................. 370
  15.4.10 Triangle-wave generation ................................................................... 372
  15.4.11 DAC channel modes .......................................................................... 373
  15.4.12 DAC channel buffer calibration .......................................................... 376
  15.4.13 Dual DAC channel conversion modes (if available) ......................... 377

15.5 DAC low-power modes .............................................................................. 382
15.6 DAC interrupts .......................................................................................... 382
15.7 DAC registers ............................................................................................ 383
  15.7.1 DAC control register (DAC_CR) ......................................................... 383
  15.7.2 DAC software trigger register (DAC_SWTRGR) ............................... 386
  15.7.3 DAC channel1 12-bit right-aligned data holding register
            (DAC_DHR12R1) ................................................................................ 387
  15.7.4 DAC channel1 12-bit left aligned data holding register
            (DAC_DHR12L1) ................................................................................ 387
  15.7.5 DAC channel1 8-bit right aligned data holding register
            (DAC_DHR8R1) ................................................................................ 388
  15.7.6 DAC channel2 12-bit right aligned data holding register
            (DAC_DHR12R2) ................................................................................ 388
  15.7.7 DAC channel2 12-bit left aligned data holding register
            (DAC_DHR12L2) ................................................................................ 389
15.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2) ................. 389
15.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD) .................. 390
15.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD) .................. 390
15.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD) .................. 391
15.7.12 DAC channel1 data output register (DAC_DOR1) ............................................. 391
15.7.13 DAC channel2 data output register (DAC_DOR2) ............................................. 392
15.7.14 DAC status register (DAC_SR) ................................................................. 392
15.7.15 DAC calibration control register (DAC_CCR) ............................................... 394
15.7.16 DAC mode control register (DAC_MCR) ....................................................... 394
15.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1) ..................... 396
15.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2) ..................... 396
15.7.19 DAC sample and hold time register (DAC_SHHR) ......................................... 397
15.7.20 DAC sample and hold refresh time register (DAC_SHRR) ................................. 397
15.7.21 DAC register map ......................................................................................... 399

16 Voltage reference buffer (VREFBUF) ................................................................. 401
16.1 Introduction ...................................................................................................... 401
16.2 VREFBUF functional description ................................................................. 401
16.3 VREFBUF registers ....................................................................................... 402
  16.3.1 VREFBUF control and status register (VREFBUF_CSR) ......................... 402
  16.3.2 VREFBUF calibration control register (VREFBUF_CCR) ....................... 403
  16.3.3 VREFBUF register map ............................................................................ 403

17 Comparator (COMP) ......................................................................................... 404
17.1 Introduction .................................................................................................... 404
17.2 COMP main features ..................................................................................... 404
17.3 COMP functional description ....................................................................... 405
  17.3.1 COMP block diagram ............................................................................. 405
  17.3.2 COMP pins and internal signals ............................................................... 405
  17.3.3 COMP reset and clocks .......................................................................... 406
  17.3.4 Comparator LOCK mechanism ............................................................. 407
  17.3.5 Window comparator ............................................................................... 407
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.3.6</td>
<td>Hysteresis</td>
<td>407</td>
</tr>
<tr>
<td>17.3.7</td>
<td>Comparator output blanking function</td>
<td>408</td>
</tr>
<tr>
<td>17.3.8</td>
<td>COMP power and speed modes</td>
<td>409</td>
</tr>
<tr>
<td>17.4</td>
<td>COMP low-power modes</td>
<td>409</td>
</tr>
<tr>
<td>17.5</td>
<td>COMP interrupts</td>
<td>409</td>
</tr>
<tr>
<td>17.6</td>
<td>COMP registers</td>
<td>410</td>
</tr>
<tr>
<td>17.6.1</td>
<td>Comparator 1 control and status register (COMP1_CSR)</td>
<td>410</td>
</tr>
<tr>
<td>17.6.2</td>
<td>Comparator 2 control and status register (COMP2_CSR)</td>
<td>412</td>
</tr>
<tr>
<td>17.6.3</td>
<td>COMP register map</td>
<td>415</td>
</tr>
<tr>
<td>18</td>
<td>True random number generator (RNG)</td>
<td>416</td>
</tr>
<tr>
<td>18.1</td>
<td>Introduction</td>
<td>416</td>
</tr>
<tr>
<td>18.2</td>
<td>RNG main features</td>
<td>416</td>
</tr>
<tr>
<td>18.3</td>
<td>RNG functional description</td>
<td>417</td>
</tr>
<tr>
<td>18.3.1</td>
<td>RNG block diagram</td>
<td>417</td>
</tr>
<tr>
<td>18.3.2</td>
<td>RNG internal signals</td>
<td>417</td>
</tr>
<tr>
<td>18.3.3</td>
<td>Random number generation</td>
<td>418</td>
</tr>
<tr>
<td>18.3.4</td>
<td>RNG initialization</td>
<td>420</td>
</tr>
<tr>
<td>18.3.5</td>
<td>RNG operation</td>
<td>421</td>
</tr>
<tr>
<td>18.3.6</td>
<td>RNG clocking</td>
<td>422</td>
</tr>
<tr>
<td>18.3.7</td>
<td>Error management</td>
<td>422</td>
</tr>
<tr>
<td>18.3.8</td>
<td>RNG low-power usage</td>
<td>423</td>
</tr>
<tr>
<td>18.4</td>
<td>RNG interrupts</td>
<td>423</td>
</tr>
<tr>
<td>18.5</td>
<td>RNG processing time</td>
<td>424</td>
</tr>
<tr>
<td>18.6</td>
<td>RNG entropy source validation</td>
<td>424</td>
</tr>
<tr>
<td>18.6.1</td>
<td>Introduction</td>
<td>424</td>
</tr>
<tr>
<td>18.6.2</td>
<td>Validation conditions</td>
<td>424</td>
</tr>
<tr>
<td>18.6.3</td>
<td>Data collection</td>
<td>424</td>
</tr>
<tr>
<td>18.7</td>
<td>RNG registers</td>
<td>425</td>
</tr>
<tr>
<td>18.7.1</td>
<td>RNG control register (RNG_CR)</td>
<td>425</td>
</tr>
<tr>
<td>18.7.2</td>
<td>RNG status register (RNG_SR)</td>
<td>426</td>
</tr>
<tr>
<td>18.7.3</td>
<td>RNG data register (RNG_DR)</td>
<td>427</td>
</tr>
<tr>
<td>18.7.4</td>
<td>RNG register map</td>
<td>428</td>
</tr>
<tr>
<td>19</td>
<td>AES hardware accelerator (AES)</td>
<td>429</td>
</tr>
<tr>
<td>19.1</td>
<td>Introduction</td>
<td>429</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>19.2</td>
<td>AES main features</td>
<td>429</td>
</tr>
<tr>
<td>19.3</td>
<td>AES implementation</td>
<td>430</td>
</tr>
<tr>
<td>19.4</td>
<td>AES functional description</td>
<td>430</td>
</tr>
<tr>
<td>19.4.1</td>
<td>AES block diagram</td>
<td>430</td>
</tr>
<tr>
<td>19.4.2</td>
<td>AES internal signals</td>
<td>430</td>
</tr>
<tr>
<td>19.4.3</td>
<td>AES cryptographic core</td>
<td>431</td>
</tr>
<tr>
<td>19.4.4</td>
<td>AES procedure to perform a cipher operation</td>
<td>436</td>
</tr>
<tr>
<td>19.4.5</td>
<td>AES decryption key preparation</td>
<td>440</td>
</tr>
<tr>
<td>19.4.6</td>
<td>AES ciphertext stealing and data padding</td>
<td>441</td>
</tr>
<tr>
<td>19.4.7</td>
<td>AES task suspend and resume</td>
<td>442</td>
</tr>
<tr>
<td>19.4.8</td>
<td>AES basic chaining modes (ECB, CBC)</td>
<td>442</td>
</tr>
<tr>
<td>19.4.9</td>
<td>AES counter (CTR) mode</td>
<td>447</td>
</tr>
<tr>
<td>19.4.10</td>
<td>AES Galois/counter mode (GCM)</td>
<td>450</td>
</tr>
<tr>
<td>19.4.11</td>
<td>AES Galois message authentication code (GMAC)</td>
<td>455</td>
</tr>
<tr>
<td>19.4.12</td>
<td>AES counter with CBC-MAC (CCM)</td>
<td>457</td>
</tr>
<tr>
<td>19.4.13</td>
<td>AES data registers and data swapping</td>
<td>463</td>
</tr>
<tr>
<td>19.4.14</td>
<td>AES key registers</td>
<td>465</td>
</tr>
<tr>
<td>19.4.15</td>
<td>AES initialization vector registers</td>
<td>465</td>
</tr>
<tr>
<td>19.4.16</td>
<td>AES DMA interface</td>
<td>465</td>
</tr>
<tr>
<td>19.4.17</td>
<td>AES error management</td>
<td>468</td>
</tr>
<tr>
<td>19.5</td>
<td>AES interrupts</td>
<td>468</td>
</tr>
<tr>
<td>19.6</td>
<td>AES processing latency</td>
<td>469</td>
</tr>
<tr>
<td>19.7</td>
<td>AES registers</td>
<td>471</td>
</tr>
<tr>
<td>19.7.1</td>
<td>AES control register (AES_CR)</td>
<td>471</td>
</tr>
<tr>
<td>19.7.2</td>
<td>AES status register (AES_SR)</td>
<td>473</td>
</tr>
<tr>
<td>19.7.3</td>
<td>AES data input register (AES_DINR)</td>
<td>475</td>
</tr>
<tr>
<td>19.7.4</td>
<td>AES data output register (AES_DOUTR)</td>
<td>476</td>
</tr>
<tr>
<td>19.7.5</td>
<td>AES key register 0 (AES_KEYR0)</td>
<td>476</td>
</tr>
<tr>
<td>19.7.6</td>
<td>AES key register 1 (AES_KEYR1)</td>
<td>477</td>
</tr>
<tr>
<td>19.7.7</td>
<td>AES key register 2 (AES_KEYR2)</td>
<td>477</td>
</tr>
<tr>
<td>19.7.8</td>
<td>AES key register 3 (AES_KEYR3)</td>
<td>477</td>
</tr>
<tr>
<td>19.7.9</td>
<td>AES initialization vector register 0 (AES_IVR0)</td>
<td>478</td>
</tr>
<tr>
<td>19.7.10</td>
<td>AES initialization vector register 1 (AES_IVR1)</td>
<td>478</td>
</tr>
<tr>
<td>19.7.11</td>
<td>AES initialization vector register 2 (AES_IVR2)</td>
<td>478</td>
</tr>
<tr>
<td>19.7.12</td>
<td>AES initialization vector register 3 (AES_IVR3)</td>
<td>479</td>
</tr>
<tr>
<td>19.7.13</td>
<td>AES key register 4 (AES_KEYR4)</td>
<td>479</td>
</tr>
</tbody>
</table>
### Advanced-control timers (TIM1) .......................... 484

#### 20.1 TIM1 introduction ................................. 484

#### 20.2 TIM1 main features ............................. 485

#### 20.3 TIM1 functional description ....................... 487

##### 20.3.1 Time-base unit ............................... 487

##### 20.3.2 Counter modes ................................ 489

##### 20.3.3 Repetition counter ........................... 500

##### 20.3.4 External trigger input ....................... 502

##### 20.3.5 Clock selection ............................... 503

##### 20.3.6 Capture/compare channels ..................... 507

##### 20.3.7 Input capture mode ............................ 510

##### 20.3.8 PWM input mode ................................ 511

##### 20.3.9 Forced output mode ........................... 512

##### 20.3.10 Output compare mode .......................... 513

##### 20.3.11 PWM mode .................................... 514

##### 20.3.12 Asymmetric PWM mode ......................... 517

##### 20.3.13 Combined PWM mode .......................... 518

##### 20.3.14 Combined 3-phase PWM mode .................. 519

##### 20.3.15 Complementary outputs and dead-time insertion ... 520

##### 20.3.16 Using the break function ..................... 522

##### 20.3.17 Bidirectional break inputs ..................... 528

##### 20.3.18 Clearing the OCxREF signal on an external event ... 530

##### 20.3.19 6-step PWM generation ....................... 531

##### 20.3.20 One-pulse mode ............................... 532

##### 20.3.21 Retriggerable one pulse mode (OPM) ............ 533

##### 20.3.22 Encoder interface mode ....................... 534

##### 20.3.23 UIF bit remapping ............................ 536

##### 20.3.24 Timer input XOR function ..................... 537

##### 20.3.25 Interfacing with Hall sensors .................. 537

##### 20.3.26 Timer synchronization ......................... 540

##### 20.3.27 ADC synchronization .......................... 544
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.3.28</td>
<td>DMA burst mode</td>
<td>544</td>
</tr>
<tr>
<td>20.3.29</td>
<td>Debug mode</td>
<td>545</td>
</tr>
<tr>
<td>20.4</td>
<td>TIM1 registers</td>
<td>546</td>
</tr>
<tr>
<td>20.4.1</td>
<td>TIM1 control register 1 (TIM1_CR1)</td>
<td>546</td>
</tr>
<tr>
<td>20.4.2</td>
<td>TIM1 control register 2 (TIM1_CR2)</td>
<td>547</td>
</tr>
<tr>
<td>20.4.3</td>
<td>TIM1 slave mode control register (TIM1_SMCR)</td>
<td>550</td>
</tr>
<tr>
<td>20.4.4</td>
<td>TIM1 DMA/interrupt enable register (TIM1_DIER)</td>
<td>552</td>
</tr>
<tr>
<td>20.4.5</td>
<td>TIM1 status register (TIM1_SR)</td>
<td>554</td>
</tr>
<tr>
<td>20.4.6</td>
<td>TIM1 event generation register (TIM1_EGR)</td>
<td>556</td>
</tr>
<tr>
<td>20.4.7</td>
<td>TIM1 capture/compare mode register 1 [alternate]</td>
<td>557</td>
</tr>
<tr>
<td></td>
<td>(TIM1_CCMR1)</td>
<td></td>
</tr>
<tr>
<td>20.4.8</td>
<td>TIM1 capture/compare mode register 1 [alternate]</td>
<td>558</td>
</tr>
<tr>
<td></td>
<td>(TIM1_CCMR1)</td>
<td></td>
</tr>
<tr>
<td>20.4.9</td>
<td>TIM1 capture/compare mode register 2 [alternate]</td>
<td>561</td>
</tr>
<tr>
<td></td>
<td>(TIM1_CCMR2)</td>
<td></td>
</tr>
<tr>
<td>20.4.10</td>
<td>TIM1 capture/compare mode register 2 [alternate]</td>
<td>562</td>
</tr>
<tr>
<td></td>
<td>(TIM1_CCMR2)</td>
<td></td>
</tr>
<tr>
<td>20.4.11</td>
<td>TIM1 capture/compare enable register</td>
<td>564</td>
</tr>
<tr>
<td></td>
<td>(TIM1_CCER)</td>
<td></td>
</tr>
<tr>
<td>20.4.12</td>
<td>TIM1 counter (TIM1_CNT)</td>
<td>567</td>
</tr>
<tr>
<td>20.4.13</td>
<td>TIM1 prescaler (TIM1_PSC)</td>
<td>567</td>
</tr>
<tr>
<td>20.4.14</td>
<td>TIM1 auto-reload register (TIM1_ARR)</td>
<td>567</td>
</tr>
<tr>
<td>20.4.15</td>
<td>TIM1 repetition counter register (TIM1_RCR)</td>
<td>568</td>
</tr>
<tr>
<td>20.4.16</td>
<td>TIM1 capture/compare register 1 (TIM1_CCR1)</td>
<td>568</td>
</tr>
<tr>
<td>20.4.17</td>
<td>TIM1 capture/compare register 2 (TIM1_CCR2)</td>
<td>569</td>
</tr>
<tr>
<td>20.4.18</td>
<td>TIM1 capture/compare register 3 (TIM1_CCR3)</td>
<td>569</td>
</tr>
<tr>
<td>20.4.19</td>
<td>TIM1 capture/compare register 4 (TIM1_CCR4)</td>
<td>570</td>
</tr>
<tr>
<td>20.4.20</td>
<td>TIM1 break and dead-time register</td>
<td>570</td>
</tr>
<tr>
<td></td>
<td>(TIM1_BDTR)</td>
<td></td>
</tr>
<tr>
<td>20.4.21</td>
<td>TIM1 DMA control register (TIM1_DCR)</td>
<td>574</td>
</tr>
<tr>
<td>20.4.22</td>
<td>TIM1 DMA address for full transfer</td>
<td>575</td>
</tr>
<tr>
<td></td>
<td>(TIM1_DMAR)</td>
<td></td>
</tr>
<tr>
<td>20.4.23</td>
<td>TIM1 option register 1 (TIM1_OR1)</td>
<td>576</td>
</tr>
<tr>
<td>20.4.24</td>
<td>TIM1 capture/compare mode register 3</td>
<td>576</td>
</tr>
<tr>
<td></td>
<td>(TIM1_CCMR3)</td>
<td></td>
</tr>
<tr>
<td>20.4.25</td>
<td>TIM1 capture/compare register 5 (TIM1_CCR5)</td>
<td>577</td>
</tr>
<tr>
<td>20.4.26</td>
<td>TIM1 capture/compare register 6 (TIM1_CCR6)</td>
<td>578</td>
</tr>
<tr>
<td>20.4.27</td>
<td>TIM1 alternate function option register 1 (TIM1_AF1)</td>
<td>579</td>
</tr>
<tr>
<td>20.4.28</td>
<td>TIM1 Alternate function register 2 (TIM1_AF2)</td>
<td>580</td>
</tr>
</tbody>
</table>
21 General-purpose timers (TIM2/TIM3) ........................................ 586
21.1 TIM2/TIM3 introduction .............................................. 586
21.2 TIM2/TIM3 main features ............................................ 586
21.3 TIM2/TIM3 functional description ................................. 588
    21.3.1 Time-base unit .............................................. 588
    21.3.2 Counter modes ............................................... 590
    21.3.3 Clock selection ............................................. 600
    21.3.4 Capture/Compare channels ................................. 604
    21.3.5 Input capture mode ........................................ 606
    21.3.6 PWM input mode ............................................ 607
    21.3.7 Forced output mode ....................................... 608
    21.3.8 Output compare mode ..................................... 608
    21.3.9 PWM mode .................................................. 609
    21.3.10 Asymmetric PWM mode ................................... 613
    21.3.11 Combined PWM mode ..................................... 613
    21.3.12 Clearing the OCxREF signal on an external event ...... 614
    21.3.13 One-pulse mode .......................................... 616
    21.3.14 Retriggerable one pulse mode (OPM) ...................... 617
    21.3.15 Encoder interface mode .................................. 618
    21.3.16 UIF bit remapping ...................................... 620
    21.3.17 Timer input XOR function ................................. 620
    21.3.18 Timers and external trigger synchronization .......... 621
    21.3.19 Timer synchronization .................................. 624
    21.3.20 DMA burst mode ......................................... 628
    21.3.21 Debug mode .............................................. 629
21.4 TIM2/TIM3 registers ................................................. 630
    21.4.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 3) .......... 630
    21.4.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 3) ........ 631
    21.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 3) 633
    21.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 3) 636
    21.4.5 TIMx status register (TIMx_SR)(x = 2 to 3) ............ 637
    21.4.6 TIMx event generation register (TIMx_EGR)(x = 2 to 3) .... 639
    21.4.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 3) 640
21.4.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)
(x = 2 to 3) ................................................................. 642
21.4.9 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)
(x = 2 to 3) ................................................................. 644
21.4.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)
(x = 2 to 3) ................................................................. 645
21.4.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 3) .... 646
21.4.12 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 3) ....................... 647
21.4.13 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 3) ....................... 648
21.4.14 TIMx prescaler (TIMx_PSC)(x = 2 to 3) ................................... 648
21.4.15 TIMx auto-reload register (TIMx_ARR)(x = 2 to 3) ..................... 649
21.4.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 3) .......... 649
21.4.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 3) ........... 650
21.4.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 3) ........... 650
21.4.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 3) ........... 651
21.4.20 TIMx DMA control register (TIMx_DCR)(x = 2 to 3) ................. 652
21.4.21 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 3) ..... 652
21.4.22 TIM2 option register 1 (TIM2_OR1) ........................................ 653
21.4.23 TIM3 option register 1 (TIM3_OR1) ........................................ 653
21.4.24 TIM2 alternate function option register 1 (TIM2_AF1) ................. 653
21.4.25 TIM3 alternate function option register 1 (TIM3_AF1) ................. 654
21.4.26 TIM2 timer input selection register (TIM2_TISEL) ..................... 654
21.4.27 TIM3 timer input selection register (TIM3_TISEL) ..................... 655
21.4.28 TIMx register map .................................................. 657

22 Basic timers (TIM6/TIM7) .............................................. 660
22.1 TIM6/TIM7 introduction .................................................. 660
22.2 TIM6/TIM7 main features ............................................... 660
22.3 TIM6/TIM7 functional description ...................................... 661
22.3.1 Time-base unit ..................................................... 661
22.3.2 Counting mode ..................................................... 663
22.3.3 UIF bit remapping .................................................. 666
22.3.4 Clock source ....................................................... 666
22.3.5 Debug mode ....................................................... 667
22.4 TIM6/TIM7 registers .................................................... 667
22.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7) ....................... 667
22.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7) ....................... 669
22.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7) ..... 669
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)</td>
<td>670</td>
</tr>
<tr>
<td>22.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7)</td>
<td>670</td>
</tr>
<tr>
<td>22.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)</td>
<td>670</td>
</tr>
<tr>
<td>22.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7)</td>
<td>671</td>
</tr>
<tr>
<td>22.4.8 TIMx auto-reload register (TIMx_ARR)(x = 6 to 7)</td>
<td>671</td>
</tr>
<tr>
<td>22.4.9 TIMx register map</td>
<td>672</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>23 General-purpose timers (TIM14)</td>
<td>673</td>
</tr>
<tr>
<td>23.1 TIM14 introduction</td>
<td>673</td>
</tr>
<tr>
<td>23.2 TIM14 main features</td>
<td>673</td>
</tr>
<tr>
<td>23.2.1 TIM14 main features</td>
<td>673</td>
</tr>
<tr>
<td>23.3 TIM14 functional description</td>
<td>675</td>
</tr>
<tr>
<td>23.3.1 Time-base unit</td>
<td>675</td>
</tr>
<tr>
<td>23.3.2 Counter modes</td>
<td>677</td>
</tr>
<tr>
<td>23.3.3 Clock selection</td>
<td>680</td>
</tr>
<tr>
<td>23.3.4 Capture/compare channels</td>
<td>681</td>
</tr>
<tr>
<td>23.3.5 Input capture mode</td>
<td>682</td>
</tr>
<tr>
<td>23.3.6 Forced output mode</td>
<td>683</td>
</tr>
<tr>
<td>23.3.7 Output compare mode</td>
<td>684</td>
</tr>
<tr>
<td>23.3.8 PWM mode</td>
<td>685</td>
</tr>
<tr>
<td>23.3.9 One-pulse mode</td>
<td>686</td>
</tr>
<tr>
<td>23.3.10 UIF bit remapping</td>
<td>686</td>
</tr>
<tr>
<td>23.3.11 Debug mode</td>
<td>687</td>
</tr>
<tr>
<td>23.4 TIM14 registers</td>
<td>688</td>
</tr>
<tr>
<td>23.4.1 TIM14 control register 1 (TIM14_CR1)</td>
<td>688</td>
</tr>
<tr>
<td>23.4.2 TIM14 Interrupt enable register (TIM14_DIER)</td>
<td>689</td>
</tr>
<tr>
<td>23.4.3 TIM14 status register (TIM14_SR)</td>
<td>689</td>
</tr>
<tr>
<td>23.4.4 TIM14 event generation register (TIM14_EGR)</td>
<td>690</td>
</tr>
<tr>
<td>23.4.5 TIM14 capture/compare mode register 1 [alternate] (TIM14_CCMR1)</td>
<td>691</td>
</tr>
<tr>
<td>23.4.6 TIM14 capture/compare mode register 1 [alternate] (TIM14_CCMR1)</td>
<td>692</td>
</tr>
<tr>
<td>23.4.7 TIM14 capture/compare enable register (TIM14_CCER)</td>
<td>694</td>
</tr>
<tr>
<td>23.4.8 TIM14 counter (TIM14_CNT)</td>
<td>695</td>
</tr>
<tr>
<td>23.4.9 TIM14 prescaler (TIM14_PSC)</td>
<td>695</td>
</tr>
<tr>
<td>23.4.10 TIM14 auto-reload register (TIM14_ARR)</td>
<td>695</td>
</tr>
<tr>
<td>23.4.11 TIM14 capture/compare register 1 (TIM14_CCR1)</td>
<td>696</td>
</tr>
<tr>
<td>23.4.12 TIM14 timer input selection register (TIM14_TISEL)</td>
<td>696</td>
</tr>
<tr>
<td>23.4.13 TIM14 register map</td>
<td>697</td>
</tr>
</tbody>
</table>
24 General-purpose timers (TIM15/TIM16/TIM17) ........................................ 699
  24.1 TIM15/TIM16/TIM17 introduction ................................................. 699
  24.2 TIM15 main features ................................................................. 699
  24.3 TIM16/TIM17 main features ...................................................... 700
  24.4 TIM15/TIM16/TIM17 functional description ................................... 703
    24.4.1 Time-base unit .............................................................. 703
    24.4.2 Counter modes ............................................................. 705
    24.4.3 Repetition counter ......................................................... 709
    24.4.4 Clock selection ........................................................... 710
    24.4.5 Capture/compare channels .............................................. 712
    24.4.6 Input capture mode ....................................................... 714
    24.4.7 PWM input mode (only for TIM15) ..................................... 715
    24.4.8 Forced output mode ..................................................... 716
    24.4.9 Output compare mode .................................................... 717
    24.4.10 PWM mode ................................................................. 718
    24.4.11 Combined PWM mode (TIM15 only) .................................. 719
    24.4.12 Complementary outputs and dead-time insertion .................... 720
    24.4.13 Using the break function .............................................. 722
    24.4.14 Bidirectional break inputs ............................................ 727
    24.4.15 One-pulse mode .......................................................... 729
    24.4.16 Retriggerable one pulse mode (OPM) (TIM15 only) ............... 731
    24.4.17 UIF bit remapping ....................................................... 731
    24.4.18 Timer input XOR function (TIM15 only) ............................... 733
    24.4.19 External trigger synchronization (TIM15 only) ...................... 734
    24.4.20 Slave mode – combined reset + trigger mode ....................... 736
    24.4.21 DMA burst mode ........................................................ 736
    24.4.22 Timer synchronization (TIM15) ........................................ 738
    24.4.23 Debug mode ............................................................... 738
  24.5 TIM15 registers .............................................................................. 739
    24.5.1 TIM15 control register 1 (TIM15_CR1) .................................... 739
    24.5.2 TIM15 control register 2 (TIM15_CR2) .................................... 740
    24.5.3 TIM15 slave mode control register (TIM15_SMCR) .................. 742
    24.5.4 TIM15 DMA/interrupt enable register (TIM15_DIER) ............... 743
    24.5.5 TIM15 status register (TIM15_SR) ....................................... 744
    24.5.6 TIM15 event generation register (TIM15_EGR) ....................... 746
    24.5.7 TIM15 capture/compare mode register 1 [alternate]
      (TIM15_CCMR1) ........................................................................ 747
### Contents

**24.5 TIM15 registers**

24.5.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) .................................................. 748
24.5.9 TIM15 capture/compare enable register (TIM15_CCER) .................................................. 751
24.5.10 TIM15 counter (TIM15_CNT) .................................................................................. 754
24.5.11 TIM15 prescaler (TIM15_PSC) .................................................................................. 754
24.5.12 TIM15 auto-reload register (TIM15_ARR) ................................................................. 754
24.5.13 TIM15 repetition counter register (TIM15_RCR) ...................................................... 755
24.5.14 TIM15 capture/compare register 1 (TIM15_CCR1) .................................................. 755
24.5.15 TIM15 capture/compare register 2 (TIM15_CCR2) .................................................. 756
24.5.16 TIM15 break and dead-time register (TIM15_BDTR) ............................................. 756
24.5.17 TIM15 DMA control register (TIM15_DCR) .......................................................... 759
24.5.18 TIM15 DMA address for full transfer (TIM15_DMAR) ........................................... 759
24.5.19 TIM15 alternate register 1 (TIM15_AF1) ................................................................. 760
24.5.20 TIM15 input selection register (TIM15_TISEL) ....................................................... 761
24.5.21 TIM15 register map ............................................................................................... 762

**24.6 TIM16/TIM17 registers**

24.6.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17) ................................................ 764
24.6.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17) ................................................ 765
24.6.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) ...................... 766
24.6.4 TIMx status register (TIMx_SR)(x = 16 to 17) ..................................................... 767
24.6.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17) ................................ 768
24.6.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17) ......................................................................................................................... 769
24.6.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17) ......................................................................................................................... 770
24.6.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) .................. 772
24.6.9 TIMx counter (TIMx_CNT)(x = 16 to 17) ............................................................. 774
24.6.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17) .......................................................... 775
24.6.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17) ........................................ 775
24.6.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) ............................. 776
24.6.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) ......................... 776
24.6.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) ....................... 777
24.6.15 TIMx DMA control register (TIMx_DCR)(x = 16 to 17) .................................... 780
24.6.16 TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17) ..................... 780
24.6.17 TIM16 alternate function register 1 (TIM16_AF1) ................................................ 781
24.6.18 TIM16 alternate function register 1 (TIM16_AF1) ................................................ 782
24.6.19 TIM16 input selection register (TIM16_TISEL) ...................................................... 784
24.6.20 TIM17 alternate function register 1 (TIM17_AF1) ................................................ 784
25 Low-power timer (LPTIM) ......................................................... 788

25.1 Introduction ................................................................. 788
25.2 LPTIM main features ...................................................... 788
25.3 LPTIM implementation ................................................... 788
25.4 LPTIM functional description ........................................... 789
  25.4.1 LPTIM block diagram .................................................. 789
  25.4.2 LPTIM pins and internal signals ................................... 789
  25.4.3 LPTIM input and trigger mapping .................................. 790
  25.4.4 LPTIM reset and clocks ............................................. 791
  25.4.5 Glitch filter ............................................................ 792
  25.4.6 Prescaler ............................................................... 792
  25.4.7 Trigger multiplexer ................................................... 793
  25.4.8 Operating mode ....................................................... 793
  25.4.9 Timeout function ...................................................... 795
  25.4.10 Waveform generation ............................................... 795
  25.4.11 Register update ..................................................... 796
  25.4.12 Counter mode ....................................................... 797
  25.4.13 Timer enable ........................................................ 797
  25.4.14 Timer counter reset ................................................. 798
  25.4.15 Encoder mode ....................................................... 798
  25.4.16 Debug mode ........................................................ 800
25.5 LPTIM low-power modes ............................................... 800
25.6 LPTIM interrupts ......................................................... 801
25.7 LPTIM registers ........................................................... 801
  25.7.1 LPTIM interrupt and status register (LPTIM_ISR) ............... 801
  25.7.2 LPTIM interrupt clear register (LPTIM_ICR) .................... 802
  25.7.3 LPTIM interrupt enable register (LPTIM_IER) ................... 803
  25.7.4 LPTIM configuration register (LPTIM_CFGR) .................... 804
  25.7.5 LPTIM control register (LPTIM_CR) ............................... 807
  25.7.6 LPTIM compare register (LPTIM_CMP) ............................ 808
  25.7.7 LPTIM autoreload register (LPTIM_ARR) ......................... 809
  25.7.8 LPTIM counter register (LPTIM_CNT) ............................. 809
  25.7.9 LPTIM configuration register 2 (LPTIM_CFGR2) ................ 810
25.7.10 LPTIM register map ........................................ 811

26 Infrared interface (IRTIM) ................................. 813

27 Independent watchdog (IWDG) ......................... 814
  27.1 Introduction ........................................ 814
  27.2 IWDG main features .................................. 814
  27.3 IWDG functional description ....................... 814
    27.3.1 IWDG block diagram .............................. 814
    27.3.2 Window option .................................. 815
    27.3.3 Hardware watchdog ............................... 816
    27.3.4 Register access protection .................... 816
    27.3.5 Debug mode .................................. 816
  27.4 IWDG registers ........................................ 817
    27.4.1 IWDG key register (IWDG_KR) .................. 817
    27.4.2 IWDG prescaler register (IWDG_PR) ............ 818
    27.4.3 IWDG reload register (IWDG_RLR) .............. 819
    27.4.4 IWDG status register (IWDG_SR) ............... 820
    27.4.5 IWDG window register (IWDG_WINR) ......... 821
    27.4.6 IWDG register map .................................. 822

28 System window watchdog (WWDG) ................. 823
  28.1 Introduction ........................................ 823
  28.2 WWDG main features .................................. 823
  28.3 WWDG functional description ....................... 823
    28.3.1 WWDG block diagram .............................. 824
    28.3.2 Enabling the watchdog .......................... 824
    28.3.3 Controlling the downcounter .................... 824
    28.3.4 Advanced watchdog interrupt feature ........... 824
    28.3.5 How to program the watchdog timeout .......... 825
    28.3.6 Debug mode .................................. 826
  28.4 WWDG registers ........................................ 826
    28.4.1 Control register (WWDG_CR) .................. 826
    28.4.2 Configuration register (WWDG_CFR) .......... 827
    28.4.3 Status register (WWDG_SR) ................... 828
    28.4.4 WWDG register map .................................. 828
29 Real-time clock (RTC) ......................................................... 829
  29.1 Introduction ......................................................... 829
  29.2 RTC main features ................................................ 829
  29.3 RTC functional description ....................................... 830
    29.3.1 RTC block diagram ......................................... 830
    29.3.2 RTC pins and internal signals ............................ 831
    29.3.3 GPIOs controlled by the RTC and TAMP .................. 832
    29.3.4 Clock and prescalers ...................................... 834
    29.3.5 Real-time clock and calendar ............................ 835
    29.3.6 Programmable alarms ...................................... 836
    29.3.7 Periodic auto-wake ......................................... 836
    29.3.8 RTC initialization and configuration .................... 837
    29.3.9 Reading the calendar ...................................... 839
    29.3.10 Resetting the RTC ........................................ 840
    29.3.11 RTC synchronization ...................................... 840
    29.3.12 RTC reference clock detection ......................... 841
    29.3.13 RTC smooth digital calibration ......................... 841
    29.3.14 Timestamp function ...................................... 843
    29.3.15 Calibration clock output ................................ 844
    29.3.16 Tamper and alarm output ................................ 844
  29.4 RTC low-power modes ............................................. 845
  29.5 RTC interrupts .................................................. 846
  29.6 RTC registers .................................................... 846
    29.6.1 RTC time register (RTC_TR) .............................. 846
    29.6.2 RTC date register (RTC_DR) .............................. 847
    29.6.3 RTC sub second register (RTC_SSR) ................. 848
    29.6.4 RTC initialization control and status register (RTC_ICSR) ......................... 848
    29.6.5 RTC prescaler register (RTC_PRER) ....................... 850
    29.6.6 RTC wakeup timer register (RTC_WUTR) .................... 851
    29.6.7 RTC control register (RTC_CR) ........................... 851
    29.6.8 RTC write protection register (RTC_WPR) .............. 854
    29.6.9 RTC calibration register (RTC_CALR) ................. 855
    29.6.10 RTC shift control register (RTC_SHIFTR) ............. 856
    29.6.11 RTC timestamp time register (RTC_TSTR) ............. 857
    29.6.12 RTC timestamp date register (RTC_TSDR) ............. 857
    29.6.13 RTC timestamp sub second register (RTC_TSSSR) ...... 858
30 Tamper and backup registers (TAMP) ........................................ 868
   30.1 Introduction ......................................................... 868
   30.2 TAMP main features ............................................. 868
   30.3 TAMP functional description .................................... 869
       30.3.1 TAMP block diagram ....................................... 869
       30.3.2 TAMP pins and internal signals ............................ 870
       30.3.3 TAMP register write protection ............................. 870
       30.3.4 Tamper detection ........................................... 871
   30.4 TAMP low-power modes ........................................... 873
   30.5 TAMP interrupts .................................................. 873
   30.6 TAMP registers .................................................... 873
       30.6.1 TAMP control register 1 (TAMP_CR1) ...................... 874
       30.6.2 TAMP control register 2 (TAMP_CR2) ...................... 875
       30.6.3 TAMP filter control register (TAMP_FLTCR) .............. 876
       30.6.4 TAMP interrupt enable register (TAMP_IER) ............. 877
       30.6.5 TAMP status register (TAMP_SR) ........................... 878
       30.6.6 TAMP masked interrupt status register (TAMP_MISR) .. 879
       30.6.7 TAMP status clear register (TAMP_SCR) .................. 880
       30.6.8 TAMP backup x register (TAMP_BKPxR) ................... 881
       30.6.9 TAMP register map ......................................... 882

31 Inter-integrated circuit (I2C) interface ................................ 883
   31.1 Introduction ....................................................... 883
   31.2 I2C main features ................................................ 883
   31.3 I2C implementation .............................................. 884
   31.4 I2C functional description ...................................... 884
       31.4.1 I2C1 block diagram ........................................ 885
### 31.4 I2C operations

- **31.4.2** I2C block diagram .................................................. 886
- **31.4.3** I2C clock requirements .............................................. 886
- **31.4.4** Mode selection .......................................................... 887
- **31.4.5** I2C initialization ...................................................... 888
- **31.4.6** Software reset .......................................................... 892
- **31.4.7** Data transfer ............................................................ 893
- **31.4.8** I2C slave mode ......................................................... 895
- **31.4.9** I2C master mode ....................................................... 904
- **31.4.10** I2C_TIMINGR register configuration examples .......... 916
- **31.4.11** SMBus specific features .......................................... 917
- **31.4.12** SMBus initialization ................................................. 920
- **31.4.13** SMBus: I2C_TIMEOUTR register configuration examples . 922
- **31.4.14** SMBus slave mode .................................................. 922
- **31.4.15** Wakeup from Stop mode on address match ............... 930
- **31.4.16** Error conditions ..................................................... 930
- **31.4.17** DMA requests ......................................................... 932
- **31.4.18** Debug mode ............................................................ 933

### 31.5 I2C low-power modes ..................................................... 933

### 31.6 I2C interrupts ............................................................. 934

### 31.7 I2C registers ............................................................. 934

#### 31.7.1 I2C control register 1 (I2C_CR1) ....................... 934
#### 31.7.2 I2C control register 2 (I2C_CR2) ....................... 937
#### 31.7.3 I2C own address 1 register (I2C_OAR1) .......... 940
#### 31.7.4 I2C own address 2 register (I2C_OAR2) .......... 941
#### 31.7.5 I2C timing register (I2C_TIMINGR) ............... 942
#### 31.7.6 I2C timeout register (I2C_TIMEOUTR) .......... 943
#### 31.7.7 I2C interrupt and status register (I2C_ISR) ...... 944
#### 31.7.8 I2C interrupt clear register (I2C_ICR) ......... 946
#### 31.7.9 I2C PEC register (I2C_PECR) ..................... 947
#### 31.7.10 I2C receive data register (I2C_RXDR) ........ 948
#### 31.7.11 I2C transmit data register (I2C_TXDR) ....... 948
#### 31.7.12 I2C register map ............................................. 949

### 32 Universal synchronous/asynchronous receiver transmitter (USART/UART) ............................................. 951

#### 32.1 USART introduction .................................................. 951
#### 32.2 USART main features ................................................ 952
32.3 USART extended features ........................................... 953
32.4 USART implementation ........................................... 953
32.5 USART functional description ................................... 954
  32.5.1 USART block diagram ....................................... 954
  32.5.2 USART signals ............................................. 955
  32.5.3 USART character description ............................... 956
  32.5.4 USART FIFOs and thresholds .............................. 958
  32.5.5 USART transmitter ........................................ 958
  32.5.6 USART receiver ........................................... 962
  32.5.7 USART baud rate generation .............................. 970
  32.5.8 Tolerance of the USART receiver to clock deviation .... 971
  32.5.9 USART Auto baud rate detection ......................... 972
  32.5.10 USART multiprocessor communication .................... 974
  32.5.11 USART Modbus communication ........................... 976
  32.5.12 USART parity control ................................... 977
  32.5.13 USART LIN (local interconnection network) mode ........ 978
  32.5.14 USART synchronous mode ................................ 980
  32.5.15 USART single-wire Half-duplex communication .......... 984
  32.5.16 USART receiver timeout ................................ 984
  32.5.17 USART Smartcard mode .................................. 985
  32.5.18 USART IrDA SIR ENDEC block .......................... 989
  32.5.19 Continuous communication using USART and DMA ......... 992
  32.5.20 RS232 Hardware flow control and RS485 Driver Enable .. 994
  32.5.21 USART low-power management .......................... 997
32.6 USART interrupts ............................................. 1000
32.7 USART registers ............................................... 1003
  32.7.1 USART control register 1 [alternate] (USART_CR1) ........ 1003
  32.7.2 USART control register 1 [alternate] (USART_CR1) ........ 1007
  32.7.3 USART control register 2 (USART_CR2) .................. 1010
  32.7.4 USART control register 3 (USART_CR3) .................. 1014
  32.7.5 USART baud rate register (USART_BRR) ................. 1019
  32.7.6 USART guard time and prescaler register (USART_GTPR) .... 1019
  32.7.7 USART receiver timeout register (USART_RTOR) ........ 1020
  32.7.8 USART request register (USART_RQR) ................... 1021
  32.7.9 USART interrupt and status register [alternate] (USART_ISR) .. 1022
  32.7.10 USART interrupt and status register [alternate] (USART_ISR) .. 1028
32.7.11 USART interrupt flag clear register (USART_ICR) ........................................ 1033
32.7.12 USART receive data register (USART_RDR) ................................................. 1035
32.7.13 USART transmit data register (USART_TDR) ................................................. 1035
32.7.14 USART prescaler register (USART_PRESC) ...................................................... 1036
32.7.15 USART register map ...................................................................................... 1037

33 Low-power universal asynchronous receiver transmitter (LPUART) .......................... 1039

33.1 LPUART introduction ......................................................................................... 1039
33.2 LPUART main features ...................................................................................... 1040
33.3 LPUART implementation .................................................................................... 1040
33.4 LPUART functional description ......................................................................... 1042
  33.4.1 LPUART block diagram ................................................................................ 1042
  33.4.2 LPUART signals ............................................................................................ 1043
  33.4.3 LPUART character description ...................................................................... 1043
  33.4.4 LPUART FIFOs and thresholds ..................................................................... 1044
  33.4.5 LPUART transmitter ...................................................................................... 1045
  33.4.6 LPUART receiver .......................................................................................... 1048
  33.4.7 LPUART baud rate generation ....................................................................... 1052
  33.4.8 Tolerance of the LPUART receiver to clock deviation .................................... 1054
  33.4.9 LPUART multiprocessor communication ...................................................... 1054
  33.4.10 LPUART parity control ................................................................................. 1056
  33.4.11 LPUART single-wire Half-duplex communication ......................................... 1057
  33.4.12 Continuous communication using DMA and LPUART ................................. 1058
  33.4.13 RS232 Hardware flow control and RS485 Driver Enable ............................... 1060
  33.4.14 LPUART low-power management ................................................................. 1063
33.5 LPUART interrupts ............................................................................................. 1066
33.6 LPUART registers .............................................................................................. 1068
  33.6.1 Control register 1 [alternate] (LPUART_CR1) .................................................... 1068
  33.6.2 Control register 1 [alternate] (LPUART_CR1) .................................................... 1071
  33.6.3 Control register 2 (LPUART_CR2) ................................................................... 1074
  33.6.4 Control register 3 (LPUART_CR3) ................................................................... 1076
  33.6.5 Baud rate register (LPUART_BRR) ................................................................. 1079
  33.6.6 Request register (LPUART_RQR) ................................................................. 1080
  33.6.7 Interrupt and status register [alternate] (LPUART_ISR) .................................... 1080
  33.6.8 Interrupt and status register [alternate] (LPUART_ISR) .................................... 1085
  33.6.9 Interrupt flag clear register (LPUART_ICR) .................................................... 1088
34 Serial peripheral interface / inter-IC sound (SPI/I2S) .................. 1093

34.1 Introduction ................................................. 1093
34.2 SPI main features ............................................. 1093
34.3 I2S main features ............................................. 1094
34.4 SPI/I2S implementation ....................................... 1094
34.5 SPI functional description .................................... 1095
  34.5.1 General description ....................................... 1095
  34.5.2 Communications between one master and one slave ........... 1096
  34.5.3 Standard multi-slave communication ........................ 1098
  34.5.4 Multi-master communication ............................... 1099
  34.5.5 Slave select (NSS) pin management ....................... 1100
  34.5.6 Communication formats ................................... 1101
  34.5.7 Configuration of SPI ...................................... 1103
  34.5.8 Procedure for enabling SPI ............................... 1104
  34.5.9 Data transmission and reception procedures ............... 1104
  34.5.10 SPI status flags ....................................... 1114
  34.5.11 SPI error flags ....................................... 1115
  34.5.12 NSS pulse mode ....................................... 1116
  34.5.13 TI mode ............................................. 1116
  34.5.14 CRC calculation ...................................... 1117
34.6 SPI interrupts ............................................ 1119
34.7 I2S functional description ................................... 1120
  34.7.1 I2S general description .................................. 1120
  34.7.2 Supported audio protocols ................................ 1121
  34.7.3 Start-up description ..................................... 1128
  34.7.4 Clock generator ....................................... 1130
  34.7.5 I2S master mode ....................................... 1133
  34.7.6 I2S slave mode ....................................... 1134
  34.7.7 I2S status flags ....................................... 1136
  34.7.8 I2S error flags ....................................... 1137
  34.7.9 DMA features ....................................... 1138
34.8 I2S interrupts ................................................................. 1138
34.9 SPI and I2S registers ................................................... 1139
  34.9.1 SPI control register 1 (SPIx_CR1) .......................... 1139
  34.9.2 SPI control register 2 (SPIx_CR2) ....................... 1141
  34.9.3 SPI status register (SPIx_SR) ................................. 1143
  34.9.4 SPI data register (SPIx_DR) ................................. 1145
  34.9.5 SPI CRC polynomial register (SPIx_CRCPR) .......... 1145
  34.9.6 SPI Rx CRC register (SPIx_RXCRCR) .................... 1145
  34.9.7 SPI Tx CRC register (SPIx_TXCRCR) .................... 1146
  34.9.8 SPIx_I2S configuration register (SPIx_I2SCFGR) ... 1146
  34.9.9 SPIx_I2S prescaler register (SPIx_I2SPR) ........... 1148
  34.9.10 SPI/I2S register map ........................................... 1150

35 USB Type-C™ / USB Power Delivery interface (UCPD) .......... 1151
  35.1 Introduction ............................................................... 1151
  35.2 UCPD main features ............................................... 1151
  35.3 UCPD implementation .............................................. 1151
  35.4 UCPD functional description ................................... 1152
    35.4.1 UCPD block diagram ........................................... 1153
    35.4.2 UCPD resets and clocks .................................. 1155
    35.4.3 Physical layer protocol ................................... 1155
    35.4.4 UCPD BMC Transmitter ..................................... 1162
    35.4.5 UCPD BMC Receiver ........................................... 1164
    35.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd) .... 1167
    35.4.7 UCPD Type-C voltage monitoring and de-bouncing .... 1167
    35.4.8 UCPD Fast Role Swap (FRS) signaling and detection ... 1167
    35.4.9 UCPD DMA Interface ........................................... 1168
    35.4.10 Wakeup from STOP ............................................. 1168
    35.4.11 UCPD programming sequences ............................ 1168
  35.5 UCPD low-power modes ............................................ 1172
  35.6 UCPD interrupts ..................................................... 1173
  35.7 UCPD registers ....................................................... 1174
    35.7.1 UCPD configuration register 1 (UCPD_CFG1) ......... 1174
    35.7.2 UCPD configuration register 2 (UCPD_CFG2) ......... 1176
    35.7.3 UCPD control register (UCPD_CR) ....................... 1177
    35.7.4 UCPD Interrupt Mask Register (UCPD_IMR) .......... 1179
35.7.5 UCPD Status Register (UCPD_SR) .......................... 1181
35.7.6 UCPD Interrupt Clear Register (UCPD_ICR) ................. 1184
35.7.7 UCPD Tx Ordered Set Type Register (UCPD_TX_ORDSET) .... 1185
35.7.8 UCPD Tx Paysize Register (UCPD_TX_PAYSZ) ............... 1186
35.7.9 UCPD Tx Data Register (UCPD_TXDR) ....................... 1187
35.7.10 UCPD Rx Ordered Set Register (UCPD_RX_ORDSET) ....... 1187
35.7.11 UCPD Rx Paysize Register (UCPD_RX_PAYSZ) ............. 1188
35.7.12 UCPD Receive Data Register (UCPD_RXDR) .................. 1189
35.7.13 UCPD Rx Ordered Set Extension Register #1
(UCPD_RX_ORDEXT1) ....................................... 1189
35.7.14 UCPD Rx Ordered Set Extension Register #2
(UCPD_RX_ORDEXT2) ....................................... 1190
35.7.15 UCPD register map ......................................... 1190

36  HDMI-CEC controller (CEC) .................................... 1193
  36.1 Introduction ............................................. 1193
  36.2 HDMI-CEC controller main features .......................... 1193
  36.3 HDMI-CEC functional description ............................ 1194
    36.3.1 HDMI-CEC pin ......................................... 1194
    36.3.2 HDMI-CEC block diagram ............................... 1194
    36.3.3 Message description ................................ 1194
    36.3.4 Bit timing ........................................... 1195
  36.4 Arbitration ............................................... 1196
    36.4.1 SFT option bit ....................................... 1197
  36.5 Error handling ............................................ 1198
    36.5.1 Bit error ........................................... 1198
    36.5.2 Message error ....................................... 1198
    36.5.3 Bit Rising Error (BRE) .............................. 1198
    36.5.4 Short Bit Period Error (SBPE) ........................ 1199
    36.5.5 Long Bit Period Error (LBPE) ........................ 1199
    36.5.6 Transmission Error Detection (TXERR) ................. 1201
  36.6 HDMI-CEC interrupts ...................................... 1202
  36.7 HDMI-CEC registers ....................................... 1203
    36.7.1 CEC control register (CEC_CR) ......................... 1203
    36.7.2 CEC configuration register (CEC_CFGCR) .............. 1204
    36.7.3 CEC Tx data register (CEC_TXDR) .................... 1207
    36.7.4 CEC Rx data register (CEC_RXDR) .................... 1207
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.7.5</td>
<td>CEC Interrupt and Status Register (CEC_ISR)</td>
<td>1207</td>
</tr>
<tr>
<td>36.7.6</td>
<td>CEC interrupt enable register (CEC_IER)</td>
<td>1209</td>
</tr>
<tr>
<td>36.7.7</td>
<td>HDMI-CEC register map</td>
<td>1211</td>
</tr>
<tr>
<td>37</td>
<td>Debug support (DBG)</td>
<td>1212</td>
</tr>
<tr>
<td>37.1</td>
<td>Overview</td>
<td>1212</td>
</tr>
<tr>
<td>37.2</td>
<td>Reference Arm documentation</td>
<td>1213</td>
</tr>
<tr>
<td>37.3</td>
<td>Pinout and debug port pins</td>
<td>1213</td>
</tr>
<tr>
<td>37.3.1</td>
<td>SWD port pins</td>
<td>1213</td>
</tr>
<tr>
<td>37.3.2</td>
<td>SW-DP pin assignment</td>
<td>1213</td>
</tr>
<tr>
<td>37.3.3</td>
<td>Internal pull-up &amp; pull-down on SWD pins</td>
<td>1214</td>
</tr>
<tr>
<td>37.4</td>
<td>ID codes and locking mechanism</td>
<td>1214</td>
</tr>
<tr>
<td>37.5</td>
<td>SWD port</td>
<td>1214</td>
</tr>
<tr>
<td>37.5.1</td>
<td>SWD protocol introduction</td>
<td>1214</td>
</tr>
<tr>
<td>37.5.2</td>
<td>SWD protocol sequence</td>
<td>1214</td>
</tr>
<tr>
<td>37.5.3</td>
<td>SW-DP state machine (reset, idle states, ID code)</td>
<td>1215</td>
</tr>
<tr>
<td>37.5.4</td>
<td>DP and AP read/write accesses</td>
<td>1216</td>
</tr>
<tr>
<td>37.5.5</td>
<td>SW-DP registers</td>
<td>1216</td>
</tr>
<tr>
<td>37.5.6</td>
<td>SW-AP registers</td>
<td>1217</td>
</tr>
<tr>
<td>37.6</td>
<td>Core debug</td>
<td>1218</td>
</tr>
<tr>
<td>37.7</td>
<td>BPU (Break Point Unit)</td>
<td>1218</td>
</tr>
<tr>
<td>37.7.1</td>
<td>BPU functionality</td>
<td>1219</td>
</tr>
<tr>
<td>37.8</td>
<td>DWT (Data Watchpoint)</td>
<td>1219</td>
</tr>
<tr>
<td>37.8.1</td>
<td>DWT functionality</td>
<td>1219</td>
</tr>
<tr>
<td>37.8.2</td>
<td>DWT Program Counter Sample Register</td>
<td>1219</td>
</tr>
<tr>
<td>37.9</td>
<td>MCU debug component (DBG)</td>
<td>1219</td>
</tr>
<tr>
<td>37.9.1</td>
<td>Debug support for low-power modes</td>
<td>1219</td>
</tr>
<tr>
<td>37.9.2</td>
<td>Debug support for timers, watchdog and I²C</td>
<td>1220</td>
</tr>
<tr>
<td>37.10</td>
<td>DBG registers</td>
<td>1220</td>
</tr>
<tr>
<td>37.10.1</td>
<td>DBG device ID code register (DBG_IDCODE)</td>
<td>1220</td>
</tr>
<tr>
<td>37.10.2</td>
<td>DBG configuration register (DBG_CR)</td>
<td>1221</td>
</tr>
<tr>
<td>37.10.3</td>
<td>DBG APB freeze register 1 (DBG_APB_FZ1)</td>
<td>1221</td>
</tr>
<tr>
<td>37.10.4</td>
<td>DBG APB freeze register 2 (DBG_APB_FZ2)</td>
<td>1223</td>
</tr>
<tr>
<td>37.10.5</td>
<td>DBG register map</td>
<td>1224</td>
</tr>
<tr>
<td>38</td>
<td>Device electronic signature</td>
<td>1226</td>
</tr>
<tr>
<td>38.1</td>
<td>Unique device ID register (96 bits)</td>
<td>1226</td>
</tr>
<tr>
<td>38.2</td>
<td>Flash memory size data register</td>
<td>1227</td>
</tr>
<tr>
<td>38.3</td>
<td>Package data register</td>
<td>1228</td>
</tr>
</tbody>
</table>

39 Revision history 1229
List of tables

Table 1. Peripherals versus products ............................................. 51
Table 2. STM32G071xx and STM32G081xx memory boundary addresses ............ 56
Table 3. STM32G071xx and STM32G041xx memory boundary addresses ............ 56
Table 4. STM32G01x peripheral register boundary addresses ................. 57
Table 5. Boot modes .............................................................. 60
Table 6. Flash memory organization .............................................. 63
Table 7. Number of wait states according to Flash memory clock (HCLK) frequency ... 64
Table 8. Page erase overview ...................................................... 67
Table 9. Mass erase overview ...................................................... 68
Table 10. Option byte format ...................................................... 72
Table 11. Organization of option bytes ........................................... 72
Table 12. Flash memory read protection status ..................................... 80
Table 13. Access status versus protection level and execution modes ............ 82
Table 14. Securable memory erase at RDP Level 1 to Level 0 change ............ 85
Table 15. FLASH interrupt requests .............................................. 86
Table 16. FLASH register map and reset values .................................. 100
Table 17. Low-power mode summary .............................................. 111
Table 18. Functionalities depending on the working mode ......................... 112
Table 19. Low-power run ........................................................... 115
Table 20. Sleep mode summary ..................................................... 116
Table 21. Low-power sleep mode summary ....................................... 118
Table 22. Stop 0 mode summary ................................................... 120
Table 23. Stop 1 mode summary ................................................... 121
Table 24. Standby mode summary .................................................. 123
Table 25. Shutdown mode summary ................................................ 125
Table 26. PWR register map and reset values ..................................... 140
Table 27. Clock source frequency .................................................. 151
Table 28. RCC register map and reset values ..................................... 192
Table 29. Port bit configuration table ............................................. 197
Table 30. GPIO register map and reset values .................................... 211
Table 31. SYSCFG register map and reset values .................................. 231
Table 32. Interconnect matrix ...................................................... 234
Table 33. DMA implementation ..................................................... 242
Table 34. DMA internal input/output signals ...................................... 243
Table 35. Programmable data width and endian behavior (when PINC = MINC = 1) ... 249
Table 36. DMA interrupt requests ................................................ 251
Table 37. DMA register map and reset values ..................................... 259
Table 38. DMAMUX instantiation ................................................... 263
Table 39. DMAMUX: assignment of multiplexer inputs to resources .......... 264
Table 40. DMAMUX: assignment of trigger inputs to resources ................. 264
Table 41. DMAMUX: assignment of synchronization inputs to resources ....... 265
Table 42. DMAMUX signals ........................................................ 267
Table 43. DMAMUX interrupts ...................................................... 271
Table 44. DMAMUX register map and reset values ................................ 276
Table 45. Vector table ............................................................... 278
Table 46. EXTI signal overview .................................................... 282
Table 47. EXTI event input configurations and register control .................. 284
Table 104. Counting direction versus encoder signals .................................................. 535
Table 105. TIM1 internal trigger connection ............................................................... 552
Table 106. Output control bits for complementary OCx and OCxN channels with break feature ................................. 566
Table 107. TIM1 register map and reset values .......................................................... 583
Table 108. Counting direction versus encoder signals ................................................ 619
Table 109. TIMx internal trigger connection .............................................................. 636
Table 110. Output control bit for standard OCx channels ........................................... 647
Table 111. TIM2/TIM3 register map and reset values .................................................. 657
Table 112. Output control bit for standard OCx channels ........................................... 672
Table 113. TIMx register map and reset values .......................................................... 677
Table 114. TIM14 register map and reset values ........................................................ 687
Table 115. Break protection disarming conditions ...................................................... 727
Table 116. TIMx Internal trigger connection ............................................................. 743
Table 117. Output control bits for complementary OCx and OCxN channels with break feature (TIM15) ....................... 753
Table 118. TIM15 register map and reset values ........................................................ 762
Table 119. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) .................... 774
Table 120. TIM16/TIM17 register map and reset values ............................................. 786
Table 121. STM32G0x1 LPTIM features ..................................................................... 788
Table 122. LPTIM input/output pins ......................................................................... 789
Table 123. LPTIM internal signals ........................................................................... 789
Table 124. LPTIM1 external trigger connection ......................................................... 790
Table 125. LPTIM2 external trigger connection ......................................................... 790
Table 126. LPTIM1 input 1 connection ..................................................................... 791
Table 127. LPTIM1 input 2 connection ..................................................................... 791
Table 128. LPTIM2 input 1 connection ..................................................................... 791
Table 129. Prescaler division ratios .......................................................................... 792
Table 130. Encoder counting scenarios .................................................................... 799
Table 131. Effect of low-power modes on the LPTIM ............................................... 800
Table 132. Interrupt events ...................................................................................... 801
Table 133. LPTIM register map and reset values ....................................................... 811
Table 134. IWDG register map and reset values ....................................................... 822
Table 135. WWDG register map and reset values ...................................................... 828
Table 136. RTC input/output pins ............................................................................ 831
Table 137. RTC internal input/output signals ............................................................. 831
Table 138. RTC configuration .................................................................................. 832
Table 139. PC13 configuration .................................................................................. 832
Table 140. RTC_OUT mapping ................................................................................ 834
Table 141. Effect of low-power modes on RTC ......................................................... 845
Table 142. Interrupt requests ................................................................................... 846
Table 143. RTC pins functionality over modes ......................................................... 845
Table 144. RTC register map and reset values .......................................................... 866
Table 145. TAMP input/output pins ........................................................................ 870
Table 146. TAMP internal input/output signals ....................................................... 870
Table 147. TAMP interconnection .......................................................................... 870
Table 148. Effect of low-power modes on TAMP ..................................................... 873
Table 149. Interrupt requests ................................................................................... 873
Table 150. TAMP register map and reset values ...................................................... 882
Table 151. STM32G0x1 I2C implementation ............................................................ 884
Table 152. Comparison of analog vs. digital filters ................................................... 888
Table 153. I2C-SMBUS specification data setup and hold times ............................... 891
<table>
<thead>
<tr>
<th>Table 176</th>
<th>Error calculation for programmed baud rates at $f_{CK} = 100$ MHz</th>
<th>1053</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 175</td>
<td>Error calculation for programmed baud rates at $lpuart_ker_ck_pres = 32,768$ KHz</td>
<td>1053</td>
</tr>
<tr>
<td>Table 174</td>
<td>LPUART features</td>
<td>. . . . 1041</td>
</tr>
<tr>
<td>Table 173</td>
<td>USART register map and reset values</td>
<td>. . . . 1037</td>
</tr>
<tr>
<td>Table 172</td>
<td>USART interrupt requests</td>
<td>. . . . 1000</td>
</tr>
<tr>
<td>Table 171</td>
<td>USART frame formats</td>
<td>. . . . 977</td>
</tr>
<tr>
<td>Table 170</td>
<td>Tolerance of the USART receiver when $BRR[3:0]$ is different from 0000</td>
<td>972</td>
</tr>
<tr>
<td>Table 169</td>
<td>Tolerance of the USART receiver when $BRR[3:0] = 0000$</td>
<td>972</td>
</tr>
<tr>
<td>Table 168</td>
<td>Noise detection from sampled data</td>
<td>. . . 968</td>
</tr>
<tr>
<td>Table 167</td>
<td>Tolerance of the USART receiver when $lpuart_ker_ck_pres = 32,768$ KHz</td>
<td>1053</td>
</tr>
<tr>
<td>Table 165</td>
<td>I2C interrupt requests</td>
<td>. . . . 934</td>
</tr>
<tr>
<td>Table 164</td>
<td>Effect of low-power modes on the I2C</td>
<td>934</td>
</tr>
<tr>
<td>Table 163</td>
<td>Examples of TIMEOUTA settings for various I2CCLK frequencies $(max\ t_{TIMEOUT} = 25$ ms)</td>
<td>922</td>
</tr>
<tr>
<td>Table 162</td>
<td>Examples of TIMEOUTB settings for various I2CCLK frequencies</td>
<td>922</td>
</tr>
<tr>
<td>Table 161</td>
<td>Examples of TIMEOUT settings for various I2CCLK frequencies $(max\ t_{IDLE} = 50$ µs)</td>
<td>922</td>
</tr>
<tr>
<td>Table 160</td>
<td>SMBUS with PEC configuration</td>
<td>. . . . 921</td>
</tr>
<tr>
<td>Table 159</td>
<td>SMBus timeout specifications</td>
<td>. . . . 919</td>
</tr>
<tr>
<td>Table 158</td>
<td>Examples of timing settings for $f_{I2CCLK} = 8$ MHz</td>
<td>916</td>
</tr>
<tr>
<td>Table 157</td>
<td>Examples of timing settings for $f_{I2CCLK} = 16$ MHz</td>
<td>916</td>
</tr>
<tr>
<td>Table 156</td>
<td>Examples of timing settings for $f_{I2CCLK} = 8$ MHz</td>
<td>917</td>
</tr>
<tr>
<td>Table 155</td>
<td>I2C-SMBUS specification clock timings</td>
<td>. . . . 906</td>
</tr>
<tr>
<td>Table 154</td>
<td>I2C configuration</td>
<td>. . . . 895</td>
</tr>
<tr>
<td>Table 153</td>
<td>SPI/I2S register map and reset values</td>
<td>. . . . 1150</td>
</tr>
<tr>
<td>Table 152</td>
<td>UCPD implementation</td>
<td>. . . . 1152</td>
</tr>
<tr>
<td>Table 151</td>
<td>UCPD pins</td>
<td>. . . . 1154</td>
</tr>
<tr>
<td>Table 150</td>
<td>UCPD internal signals</td>
<td>. . . . 1154</td>
</tr>
<tr>
<td>Table 149</td>
<td>4b5b Symbol Encoding Table</td>
<td>. . . . 1156</td>
</tr>
<tr>
<td>Table 148</td>
<td>Data size</td>
<td>. . . . 1158</td>
</tr>
<tr>
<td>Table 147</td>
<td>Validation of ordered sets</td>
<td>. . . . 1158</td>
</tr>
<tr>
<td>Table 146</td>
<td>Ordered sets</td>
<td>. . . . 1158</td>
</tr>
<tr>
<td>Table 145</td>
<td>Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx</td>
<td>. . . . 1169</td>
</tr>
<tr>
<td>Table 144</td>
<td>Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)</td>
<td>. . . . 1170</td>
</tr>
<tr>
<td>Table 143</td>
<td>Effect of low power modes on the UCPD</td>
<td>. . . . 1172</td>
</tr>
<tr>
<td>Table 142</td>
<td>UCPD interrupt requests</td>
<td>. . . . 1173</td>
</tr>
<tr>
<td>Table 141</td>
<td>UCPD register map and reset values</td>
<td>. . . . 1190</td>
</tr>
<tr>
<td>Table 140</td>
<td>HDMI pin</td>
<td>. . . . 1194</td>
</tr>
<tr>
<td>Table 139</td>
<td>Error handling timing parameters</td>
<td>. . . . 1200</td>
</tr>
<tr>
<td>Table 138</td>
<td>TXERR timing parameters</td>
<td>. . . . 1201</td>
</tr>
<tr>
<td>Table 137</td>
<td>HDMI-CEC interrupts</td>
<td>. . . . 1202</td>
</tr>
<tr>
<td>Table 136</td>
<td>HDMI-CEC register map and reset values</td>
<td>. . . . 1211</td>
</tr>
<tr>
<td>Table 135</td>
<td>SW debug port pins</td>
<td>. . . . 1213</td>
</tr>
<tr>
<td>Table 134</td>
<td>Packet request (8-bits)</td>
<td>. . . . 1215</td>
</tr>
</tbody>
</table>
List of tables

Table 205. ACK response (3 bits) ................................................................. 1215
Table 206. DATA transfer (33 bits) ............................................................... 1215
Table 207. SW-DP registers ................................................................. 1216
Table 208. 32-bit debug port registers addressed through the shifted value A[3:2] ............ 1217
Table 209. Core debug registers ............................................................... 1218
Table 210. DEV_ID and REV_ID field values ............................................ 1220
Table 211. DBG register map and reset values ............................................ 1224
Table 212. Document revision history ......................................................... 1229
List of figures

Figure 1. System architecture ................................................................. 52
Figure 2. Memory map ..................................................................... 55
Figure 3. Changing read protection (RDP) level ........................................ 82
Figure 4. Example of disabling core debug access ..................................... 86
Figure 5. Power supply overview .............................................................. 103
Figure 6. POR, PDR, and BOR thresholds ............................................... 107
Figure 7. PVD thresholds .................................................................... 108
Figure 8. Low-power modes state diagram ............................................. 110
Figure 9. Simplified diagram of the reset circuit ....................................... 143
Figure 10. Clock tree .......................................................................... 147
Figure 11. HSE/ LSE clock sources .......................................................... 148
Figure 12. Frequency measurement with TIM14 in capture mode ................. 155
Figure 13. Frequency measurement with TIM16 in capture mode ............... 155
Figure 14. Frequency measurement with TIM17 in capture mode ............... 156
Figure 15. Basic structure of an I/O port bit ............................................ 197
Figure 16. Input floating/pull up/pull down configurations ......................... 201
Figure 17. Output configuration ............................................................... 202
Figure 18. Alternate function configuration ............................................. 203
Figure 19. High impedance-analog configuration ..................................... 203
Figure 20. DMA block diagram ............................................................... 242
Figure 21. DMAMUX block diagram ......................................................... 266
Figure 22. Synchronization mode of the DMAMUX request line multiplexer channel ......................................................... 269
Figure 23. Event generation of the DMA request line multiplexer channel .... 269
Figure 24. EXTI block diagram ............................................................... 282
Figure 25. Configurable event trigger logic CPU wakeup ......................... 284
Figure 26. Direct event trigger logic CPU wakeup .................................... 285
Figure 27. EXTI GPIO mux ................................................................... 286
Figure 28. CRC calculation unit block diagram ....................................... 297
Figure 29. ADC block diagram ............................................................... 304
Figure 30. ADC calibration ................................................................. 307
Figure 31. Calibration factor forcing ......................................................... 307
Figure 32. Enabling/disabling the ADC .................................................. 308
Figure 33. ADC clock scheme ............................................................... 309
Figure 34. ADC connectivity ............................................................... 311
Figure 35. Analog to digital conversion time ........................................... 315
Figure 36. ADC conversion timings ........................................................ 316
Figure 37. Stopping an ongoing conversion ............................................ 316
Figure 38. Single conversions of a sequence, software trigger ..................... 320
Figure 39. Continuous conversion of a sequence, software trigger ............. 321
Figure 40. Single conversions of a sequence, hardware trigger .................. 321
Figure 41. Continuous conversions of a sequence, hardware trigger ........... 322
Figure 42. Data alignment and resolution (oversampling disabled: OVSE = 0) ................................................................. 323
Figure 43. Example of overrun (OVR) ..................................................... 324
Figure 44. Wait mode conversion (continuous mode, software trigger) ........ 326
Figure 45. Behavior with WAIT=0, AUTOFF=1 ........................................ 327
Figure 46. Behavior with WAIT=1, AUTOFF=1 ........................................ 328
Figure 47. Analog watchdog guarded area .............................................. 329
Figure 48. ADC_AWDx_OUT signal generation ...................................... 330
Figure 49. ADC_AWDx_OUT signal generation (AWDx flag not cleared by software) ............ 331
Figure 50. ADC_AWDx_OUT signal generation (on a single channel) .......................... 331
Figure 51. Analog watchdog threshold update .......................................................... 332
Figure 52. 20-bit to 16-bit result truncation ................................................................. 332
Figure 53. Numerical example with 5-bits shift and rounding ....................................... 333
Figure 54. Triggered oversampling mode (TOVS bit = 1) ............................................ 335
Figure 55. Temperature sensor and VREFINT channel block diagram ......................... 336
Figure 56. VBAT channel block diagram ...................................................................... 338
Figure 57. Dual-channel DAC block diagram ............................................................... 345
Figure 58. Data registers in single DAC channel mode .................................................. 366
Figure 59. Data registers in dual DAC channel mode ..................................................... 368
Figure 60. Timing diagram for conversion with trigger disabled TEN = 0 ...................... 368
Figure 61. DAC LFSR register calculation algorithm ..................................................... 371
Figure 62. DAC conversion (SW trigger enabled) with LFSR wave generation .......... 371
Figure 63. DAC triangle wave generation ................................................................... 372
Figure 64. DAC conversion (SW trigger enabled) with triangle wave generation .......... 372
Figure 65. DAC Sample and Hold mode phase diagram ............................................... 375
Figure 66. Comparator block diagram ........................................................................ 405
Figure 67. Window mode ............................................................................................. 407
Figure 68. Comparator hysteresis ................................................................................ 408
Figure 69. Comparator output blanking ......................................................................... 408
Figure 70. RNG block diagram ..................................................................................... 417
Figure 71. Entropy source model ................................................................................... 418
Figure 72. RNG initialization overview ......................................................................... 421
Figure 73. AES block diagram ..................................................................................... 430
Figure 74. ECB encryption and decryption principle ...................................................... 432
Figure 75. CBC encryption and decryption principle ...................................................... 433
Figure 76. CTR encryption and decryption principle ...................................................... 434
Figure 77. GCM encryption and authentication principle ............................................... 435
Figure 78. GMAC authentication principle ..................................................................... 435
Figure 79. CCM encryption and authentication principle ............................................... 436
Figure 80. STM32 cryptolib AES flowchart examples ................................................... 437
Figure 81. STM32 cryptolib AES flowchart examples (continued) ............................... 438
Figure 82. Encryption key derivation for ECB/CBC decryption (Mode 2) ...................... 441
Figure 83. Example of suspend mode management ....................................................... 442
Figure 84. ECB encryption ............................................................................................ 443
Figure 85. ECB decryption ............................................................................................ 443
Figure 86. CBC encryption ........................................................................................... 444
Figure 87. CBC decryption ............................................................................................ 444
Figure 88. ECB/CBC encryption (Mode 1) .................................................................... 445
Figure 89. ECB/CBC decryption (Mode 3) .................................................................... 446
Figure 90. Message construction in CTR mode ............................................................. 448
Figure 91. CTR encryption ............................................................................................ 448
Figure 92. CTR decryption ............................................................................................ 449
Figure 93. Message construction in GCM ..................................................................... 449
Figure 94. GCM authenticated encryption ..................................................................... 452
Figure 95. Message construction in GMAC mode ........................................................... 456
Figure 96. GMAC authentication mode ......................................................................... 456
Figure 97. Message construction in CCM mode ............................................................. 457
Figure 98. CCM mode authenticated encryption ............................................................. 459
Figure 99. 128-bit block construction with respect to data swap .................................... 464
Figure 100. DMA transfer of a 128-bit data block during input phase ......................... 466
Figure 101. DMA transfer of a 128-bit data block during output phase
Figure 102. AES interrupt signal generation
Figure 103. Advanced-control timer block diagram
Figure 104. Counter timing diagram with prescaler division change from 1 to 2
Figure 105. Counter timing diagram with prescaler division change from 1 to 4
Figure 106. Counter timing diagram, internal clock divided by 1
Figure 107. Counter timing diagram, internal clock divided by 2
Figure 108. Counter timing diagram, internal clock divided by 4
Figure 109. Counter timing diagram, internal clock divided by N
Figure 110. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)
Figure 111. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)
Figure 112. Counter timing diagram, internal clock divided by 1
Figure 113. Counter timing diagram, internal clock divided by 2
Figure 114. Counter timing diagram, internal clock divided by 4
Figure 115. Counter timing diagram, internal clock divided by N
Figure 116. Counter timing diagram, update event when repetition counter is not used
Figure 117. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6
Figure 118. Counter timing diagram, internal clock divided by 2
Figure 119. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36
Figure 120. Counter timing diagram, internal clock divided by N
Figure 121. Counter timing diagram, update event with ARPE=1 (counter underflow)
Figure 122. Counter timing diagram, Update event with ARPE=1 (counter overflow)
Figure 123. Update rate examples depending on mode and TIMx_RCR register settings
Figure 124. External trigger input block
Figure 125. TIM1 ETR input circuitry
Figure 126. Control circuit in normal mode, internal clock divided by 1
Figure 127. T12 external clock connection example
Figure 128. Control circuit in external clock mode 1
Figure 129. External trigger input block
Figure 130. Control circuit in external clock mode 2
Figure 131. Capture/compare channel (example: channel 1 input stage)
Figure 132. Capture/compare channel 1 main circuit
Figure 133. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)
Figure 134. Output stage of capture/compare channel (channel 4)
Figure 135. Output stage of capture/compare channel (channel 5, idem ch. 6)
Figure 136. PWM input mode timing
Figure 137. Output compare mode, toggle on OC1
Figure 138. Edge-aligned PWM waveforms (ARR=8)
Figure 139. Center-aligned PWM waveforms (ARR=8)
Figure 140. Generation of 2 phase-shifted PWM signals with 50% duty cycle
Figure 141. Combined PWM mode on channel 1 and 3
Figure 142. 3-phase combined PWM signals with multiple trigger pulses per period
Figure 143. Complementary output with dead-time insertion
Figure 144. Dead-time waveforms with delay greater than the negative pulse
Figure 145. Dead-time waveforms with delay greater than the positive pulse
Figure 146. Break and Break2 circuitry overview
Figure 147. Various output behavior in response to a break event on BRK (OSSI = 1)
Figure 148. PWM output state following BRK and BRK2 pins assertion (OSSI=1)
Figure 149. PWM output state following BRK assertion (OSSI=0)
Figure 150. Output redirection (BRK2 request not represented)
Figure 151. Clearing TIMx OCxREF
Figure 152. 6-step generation, COM example (OSSR=1)
Figure 153. Example of one pulse mode. ............................................................... 532
Figure 154. Retriggerable one pulse mode ....................................................... 534
Figure 155. Example of counter operation in encoder interface mode. .......... 535
Figure 156. Example of encoder interface mode with TI1FP1 polarity inverted. 536
Figure 157. Measuring time interval between edges on 3 signals .................... 537
Figure 158. Example of Hall sensor interface ................................................... 539
Figure 159. Control circuit in reset mode ......................................................... 540
Figure 160. Control circuit in Gated mode ....................................................... 541
Figure 161. Control circuit in trigger mode ..................................................... 542
Figure 162. Control circuit in external clock mode 2 + trigger mode ............... 543
Figure 163. General-purpose timer block diagram ......................................... 587
Figure 164. Counter timing diagram with prescaler division change from 1 to 2 . 589
Figure 165. Counter timing diagram with prescaler division change from 1 to 4 . 589
Figure 166. Counter timing diagram, internal clock divided by 1 ....................... 590
Figure 167. Counter timing diagram, internal clock divided by 2 ....................... 591
Figure 168. Counter timing diagram, internal clock divided by 4 ....................... 591
Figure 169. Counter timing diagram, internal clock divided by N ..................... 592
Figure 170. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded) .............................................................. 592
Figure 171. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded) .............................................................. 593
Figure 172. Counter timing diagram, internal clock divided by 1 ....................... 594
Figure 173. Counter timing diagram, internal clock divided by 2 ....................... 594
Figure 174. Counter timing diagram, internal clock divided by 4 ....................... 595
Figure 175. Counter timing diagram, internal clock divided by N ..................... 595
Figure 176. Counter timing diagram, Update event when repetition counter is not used .............................................................. 596
Figure 177. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6 .............................................................. 597
Figure 178. Counter timing diagram, internal clock divided by 2 ....................... 598
Figure 179. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 .............................................................. 598
Figure 180. Counter timing diagram, internal clock divided by N ..................... 599
Figure 181. Counter timing diagram, Update event with ARPE=1 (counter underflow) .............................................................. 599
Figure 182. Counter timing diagram, Update event with ARPE=1 (counter overflow) .............................................................. 600
Figure 183. Control circuit in normal mode, internal clock divided by 1 ............ 601
Figure 184. TI2 external clock connection example ........................................ 601
Figure 185. Control circuit in external clock mode 1 ....................................... 602
Figure 186. External trigger input block .......................................................... 603
Figure 187. Control circuit in external clock mode 2 ....................................... 604
Figure 188. Capture/Compare channel (example: channel 1 input stage) ............ 604
Figure 189. Capture/Compare channel 1 main circuit .................................... 605
Figure 190. Output stage of Capture/Compare channel (channel 1) .................... 605
Figure 191. PWM input mode timing ............................................................. 607
Figure 192. Output compare mode, toggle on OC1 ......................................... 609
Figure 193. Edge-aligned PWM waveforms (ARR=8) .................................... 610
Figure 194. Center-aligned PWM waveforms (ARR=8) .................................. 612
Figure 195. Generation of 2 phase-shifted PWM signals with 50% duty cycle .... 613
Figure 196. Combined PWM mode on channels 1 and 3 .................................. 614
Figure 197. Clearing TIMx OCxREF .............................................................. 615
Figure 198. Example of one-pulse mode ......................................................... 616
Figure 199. Retriggerable one-pulse mode ...................................................... 618
Figure 200. Example of counter operation in encoder interface mode ............... 619
Figure 201. Example of encoder interface mode with TI1FP1 polarity inverted .... 620
Figure 202. Control circuit in reset mode ....................................................... 621
Figure 203. Control circuit in gated mode ...................................................... 622
List of figures

Figure 250. Control circuit in external clock mode 1 .................................................. 712
Figure 251. Capture/compare channel (example: channel 1 input stage) ...................... 713
Figure 252. Capture/compare channel 1 main circuit .................................................. 713
Figure 253. Output stage of capture/compare channel (channel 1) .............................. 714
Figure 254. Output stage of capture/compare channel (channel 2 for TIM15) .............. 714
Figure 255. PWM input mode timing ............................................................................. 716
Figure 256. Output compare mode, toggle on OC1 ....................................................... 718
Figure 257. Edge-aligned PWM waveforms (ARR=8) ............................................... 719
Figure 258. Combined PWM mode on channel 1 and 2 ............................................ 720
Figure 259. Complementary output with dead-time insertion ..................................... 721
Figure 260. Dead-time waveforms with delay greater than the negative pulse .......... 721
Figure 261. Dead-time waveforms with delay greater than the positive pulse ......... 722
Figure 262. Break circuitry overview ............................................................................ 724
Figure 263. Output behavior in response to a break .................................................... 726
Figure 264. Output redirection ..................................................................................... 728
Figure 265. Example of one pulse mode ..................................................................... 730
Figure 266. Retriggerable one pulse mode .................................................................. 731
Figure 267. Measuring time interval between edges on 2 signals .............................. 733
Figure 268. Control circuit in reset mode ................................................................... 734
Figure 269. Control circuit in gated mode ................................................................... 735
Figure 270. Control circuit in trigger mode ................................................................ 736
Figure 271. Low-power timer block diagram (LPTIM1 and LPTIM2(1)) .................... 789
Figure 272. Glitch filter timing diagram ..................................................................... 792
Figure 273. LPTIM output waveform, single counting mode configuration .............. 794
Figure 274. LPTIM output waveform, Continuous counting mode configuration and Set-once mode activated (WAVE bit is set) ................................................................. 794
Figure 275. LPTIM output waveform, Continuous counting mode configuration ....... 795
Figure 276. Waveform generation ............................................................................ 796
Figure 277. Encoder mode counting sequence ......................................................... 800
Figure 278. IRTIM internal hardware connections ..................................................... 813
Figure 279. Independent watchdog block diagram .................................................... 814
Figure 280. Watchdog block diagram ....................................................................... 824
Figure 281. Window watchdog timing diagram ........................................................... 825
Figure 282. RTC block diagram ............................................................................... 830
Figure 283. TAMP block diagram ......................................................................... 869
Figure 284. I2C1 block diagram .............................................................................. 885
Figure 285. I2C2 block diagram .............................................................................. 886
Figure 286. I2C bus protocol .................................................................................... 888
Figure 287. Setup and hold timings .......................................................................... 889
Figure 288. I2C initialization flowchart ..................................................................... 892
Figure 289. Data protocol ......................................................................................... 893
Figure 290. Data transmission .................................................................................. 894
Figure 291. Slave initialization flowchart ................................................................. 897
Figure 292. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=0 .... 899
Figure 293. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=1 ... 900
Figure 294. Transfer bus diagrams for I2C slave transmitter ................................... 901
Figure 295. Transfer sequence flowchart for slave receiver with NOSTRETCH=0 ..... 902
Figure 296. Transfer sequence flowchart for slave receiver with NOSTRETCH=1 ... 903
Figure 297. Transfer bus diagrams for I2C slave receiver ....................................... 903
Figure 298. Master clock generation ....................................................................... 905
Figure 299. Master initialization flowchart ............................................................... 907
Figure 300. 10-bit address read access with HEAD10R=0 ..................................... 907
Figure 301. 10-bit address read access with HEAD10R = 1 .......................................................... 908
Figure 302. Transfer sequence flowchart for I2C master transmitter for N ≤ 255 bytes ...................... 909
Figure 303. Transfer sequence flowchart for I2C master transmitter for N > 255 bytes ....................... 910
Figure 304. Transfer bus diagrams for I2C master transmitter .......................................................... 911
Figure 305. Transfer sequence flowchart for I2C master receiver for N ≤ 255 bytes .......................... 913
Figure 306. Transfer sequence flowchart for I2C master receiver for N > 255 bytes ......................... 914
Figure 307. Transfer bus diagrams for I2C master receiver ............................................................. 915
Figure 308. Timeout intervals for t LOW:SEXT, t LOW:MEXT .......................................................... 919
Figure 309. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC ......................... 923
Figure 310. Transfer bus diagrams for SMBus slave transmitter (SBC = 1) ...................................... 924
Figure 311. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC ........................... 925
Figure 312. Bus transfer diagrams for SMBus slave receiver (SBC = 1) ........................................... 926
Figure 313. Bus transfer diagrams for SMBus master transmitter ....................................................... 927
Figure 314. Bus transfer diagrams for SMBus master receiver .......................................................... 929
Figure 315. USART block diagram .................................................................................................. 954
Figure 316. Word length programming .............................................................................................. 957
Figure 317. Configurable stop bits ..................................................................................................... 959
Figure 318. TC/TXE behavior when transmitting ................................................................................. 962
Figure 319. Start bit detection when oversampling by 16 or 8 .............................................................. 963
Figure 320. uart_ker_ck clock divider block diagram ......................................................................... 966
Figure 321. Data sampling when oversampling by 16 ....................................................................... 968
Figure 322. Data sampling when oversampling by 8 ....................................................................... 968
Figure 323. Mute mode using Idle line detection .............................................................................. 975
Figure 324. Mute mode using address mark detection ....................................................................... 976
Figure 325. Break detection in LIN mode (11-bit break length - LBDL bit is set) .............................. 979
Figure 326. Break detection in LIN mode vs. Framing error detection ............................................. 980
Figure 327. USART example of synchronous master transmission .................................................... 981
Figure 328. USART data clock timing diagram in synchronous master mode (M bits = 0 0) ............ 981
Figure 329. USART data clock timing diagram in synchronous master mode (M bits = '01') ........... 982
Figure 330. USART data clock timing diagram in synchronous slave mode (M bits = 0 0) ............ 983
Figure 331. ISO 7816-3 asynchronous protocol ............................................................................... 985
Figure 332. Parity error detection using the 1.5 stop bits ..................................................................... 987
Figure 333. IrDA SIR ENDEC block diagram .................................................................................... 991
Figure 334. IrDA data modulation (3/16) - Normal mode ................................................................. 991
Figure 335. Transmission using DMA .............................................................................................. 993
Figure 336. Reception using DMA ..................................................................................................... 994
Figure 337. Hardware flow control between 2 USARTs ...................................................................... 994
Figure 338. RS232 RTS flow control ................................................................................................. 995
Figure 339. RS232 CTS flow control ................................................................................................. 996
Figure 340. Wakeup event verified (wakeup event = address match, FIFO disabled) ...................... 999
Figure 341. Wakeup event not verified (wakeup event = address match, FIFO disabled) .................. 999
Figure 342. LPUART block diagram ................................................................................................. 1042
Figure 343. LPUART word length programming ............................................................................ 1044
Figure 344. Configurable stop bits .................................................................................................... 1046
Figure 345. TC/TXE behavior when transmitting ............................................................................. 1048
Figure 346. lpuart_ker_ck clock divider block diagram ................................................................. 1051
Figure 347. Mute mode using Idle line detection .............................................................................. 1055
Figure 348. Mute mode using address mark detection ....................................................................... 1056
Figure 349. Transmission using DMA ................................................................. 1059
Figure 350. Reception using DMA ................................................................. 1060
Figure 351. Hardware flow control between 2 LPUARTs ................................. 1060
Figure 352. RS232 RTS flow control ............................................................ 1061
Figure 353. RS232 CTS flow control ............................................................. 1062
Figure 354. Wakeup event verified (wakeup event = address match, FIFO disabled) ......................................................... 1098
Figure 355. Wakeup event not verified (wakeup event = address match, FIFO disabled) ......................................................... 1065
Figure 356. SPI block diagram ..................................................................... 1095
Figure 357. Full-duplex single master/ single slave application ....................... 1096
Figure 358. Half-duplex single master/ single slave application ....................... 1097
Figure 359. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode) ......................................................... 1098
Figure 360. Master and three independent slaves ........................................... 1099
Figure 361. Multi-master application ............................................................ 1100
Figure 362. Hardware/software slave select management ............................. 1101
Figure 363. Data clock timing diagram ......................................................... 1102
Figure 364. Data alignment when data length is not equal to 8-bit or 16-bit .... 1103
Figure 365. Packing data in FIFO for transmission and reception .................. 1107
Figure 366. Master full-duplex communication ............................................. 1110
Figure 367. Slave full-duplex communication ............................................... 1111
Figure 368. Master full-duplex communication with CRC .............................. 1112
Figure 369. Master full-duplex communication in packed mode ................... 1113
Figure 370. NSSP pulse generation in Motorola SPI master mode .................. 1116
Figure 371. TI mode transfer ....................................................................... 1117
Figure 372. I2S block diagram ..................................................................... 1120
Figure 373. I2S Philips protocol waveforms (16/32-bit full accuracy) .............. 1122
Figure 374. I2S Philips standard waveforms (24-bit frame) ............................. 1122
Figure 375. Transmitting 0x8EAA33 ............................................................. 1123
Figure 376. Receiving 0x8EAA33 ................................................................. 1123
Figure 377. I2S Philips standard (16-bit extended to 32-bit packet frame) ....... 1123
Figure 378. Example of 16-bit data frame extended to 32-bit channel frame .... 1123
Figure 379. MSB Justified 16-bit or 32-bit full-accuracy length ....................... 1124
Figure 380. MSB justified 24-bit frame length ............................................... 1124
Figure 381. MSB justified 16-bit extended to 32-bit packet frame ................... 1125
Figure 382. LSB justified 16-bit or 32-bit full-accuracy ................................. 1125
Figure 383. LSB justified 24-bit frame length ............................................... 1125
Figure 384. Operations required to transmit 0x3478AE ................................ 1126
Figure 385. Operations required to receive 0x3478AE .................................. 1126
Figure 386. LSB justified 16-bit extended to 32-bit packet frame ................... 1126
Figure 387. Example of 16-bit data frame extended to 32-bit channel frame .... 1127
Figure 388. PCM standard waveforms (16-bit) ............................................. 1127
Figure 389. PCM standard waveforms (16-bit extended to 32-bit packet frame) .......... 1128
Figure 390. Start sequence in master mode .................................................... 1129
Figure 391. Audio sampling frequency definition ......................................... 1130
Figure 392. I2S clock generator architecture ............................................... 1130
Figure 393. UCPD block diagram ................................................................ 1153
Figure 394. Clock division and timing elements ........................................... 1155
Figure 395. K-code transmission ................................................................. 1157
Figure 396. Transmit order for various sizes of data ..................................... 1159
Figure 397. Packet format ........................................................................... 1159
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>398</td>
<td>Line format of Hard Reset.</td>
<td>1160</td>
</tr>
<tr>
<td>399</td>
<td>Line format of Cable Reset.</td>
<td>1160</td>
</tr>
<tr>
<td>400</td>
<td>BIST Test Data Frame</td>
<td>1161</td>
</tr>
<tr>
<td>401</td>
<td>BIST Carrier Mode 2 Frame</td>
<td>1162</td>
</tr>
<tr>
<td>402</td>
<td>UCPD BMC Transmitter architecture</td>
<td>1163</td>
</tr>
<tr>
<td>403</td>
<td>UCPD BMC Receiver architecture</td>
<td>1165</td>
</tr>
<tr>
<td>404</td>
<td>HDMI-CEC block diagram</td>
<td>1194</td>
</tr>
<tr>
<td>405</td>
<td>Message structure</td>
<td>1195</td>
</tr>
<tr>
<td>406</td>
<td>Blocks</td>
<td>1195</td>
</tr>
<tr>
<td>407</td>
<td>Bit timings</td>
<td>1196</td>
</tr>
<tr>
<td>408</td>
<td>Signal free time</td>
<td>1196</td>
</tr>
<tr>
<td>409</td>
<td>Arbitration phase</td>
<td>1197</td>
</tr>
<tr>
<td>410</td>
<td>SFT of three nominal bit periods</td>
<td>1197</td>
</tr>
<tr>
<td>411</td>
<td>Error bit timing</td>
<td>1198</td>
</tr>
<tr>
<td>412</td>
<td>Error handling</td>
<td>1199</td>
</tr>
<tr>
<td>413</td>
<td>TXERR detection</td>
<td>1201</td>
</tr>
<tr>
<td>414</td>
<td>Block diagram of STM32G0x1 MCU and Cortex®-M0+-level debug support</td>
<td>1212</td>
</tr>
</tbody>
</table>
1 Documentation conventions

1.1 General information

The STM32G0x1 devices have an Arm® Cortex®-M0+ core.

1.2 List of abbreviations for registers

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

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

---

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

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

- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **SWD-DP (SWD DEBUG PORT)**: SWD-DP provides a 2-pin (clock and data) interface based on the Serial Wire Debug (SWD) protocol. Please refer to the Cortex®-M0+ technical reference manual.
- **IAP (in-application programming)**: IAP is the ability to re-program the Flash memory of a microcontroller while the user program is running.
- **ICP (in-circuit programming)**: ICP is the ability to program the Flash memory of a microcontroller using the SWD protocol or the bootloader while the device is mounted on the user application board.
- **Option bytes**: product configuration bits stored in the Flash memory.
- **OBL**: option byte loader.
- **AHB**: advanced high-performance bus.
- **APB**: advanced peripheral bus.

1.4 Availability of peripherals

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

The following table shows per-product availability of peripherals that are not common to all STM32G0x1 products.

<table>
<thead>
<tr>
<th>Feature</th>
<th>STM32G031</th>
<th>STM32G041</th>
<th>STM32G071</th>
<th>STM32G081</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNG</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>AES</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DAC</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COMP</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TIM6 and TIM7</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TIM15</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>USART3, USART4</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UCPD</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CEC</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
2 Memory and bus architecture

2.1 System architecture

The main system consists of:
- Two masters:
  - Cortex®-M0+ core
  - General-purpose DMA
- Three slaves:
  - Internal SRAM
  - Internal Flash memory
  - AHB with AHB-to-APB bridge that connects all the APB peripherals

These are interconnected using a multilayer AHB bus architecture as shown in Figure 1.

The RNG and AES functions are only supported on STM32G081xx and STM32G041xx devices.

Figure 1. System architecture

System bus (S-bus)

This bus connects the system bus of the Cortex®-M0+ core (peripheral bus) to a bus matrix that manages the arbitration between the core and the DMA.
DMA bus
This bus connects the AHB master interface of the DMA to the bus matrix that manages the access of CPU and DMA to SRAM, Flash memory and AHB/APB peripherals.

Bus matrix
The bus matrix manages the access arbitration between the core system bus and the DMA master bus. The arbitration uses a Round Robin algorithm. The bus matrix is composed of masters (CPU, DMA) and slaves (Flash memory interface, SRAM and AHB-to-APB bridge). AHB peripherals are connected on system bus through the bus matrix to allow DMA access.

AHB-to-APB bridge (APB)
The AHB-to-APB bridge provides full synchronous connections between the AHB and the APB bus.

Refer to Section 2.2: Memory organization for the address mapping of the peripherals connected to this bridge.

After each device reset, all peripheral clocks are disabled (except for the SRAM and Flash memory). Before using a peripheral its clock in the RCC_AHBENR, RCC_APBENRx or RCC_IOPENR register must first be enabled.

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

2.2.1 Introduction

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

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

The addressable memory space is divided into eight main blocks, of 512 Mbytes each.
2.2.2 Memory map and register boundary addresses

Figure 2. Memory map

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

1. Applies to STM32G071xx and STM32G081xx. For STM32G031xx and STM32G041xx, the value is 0x0000 FFFF.
2. Applies to STM32G071xx and STM32G081xx. For STM32G031xx and STM32G041xx, the value is 0x0800 FFFF.
3. Depends on boot configuration.
### Table 2. STM32G071xx and STM32G081xx memory boundary addresses

<table>
<thead>
<tr>
<th>Type</th>
<th>Boundary address</th>
<th>Size</th>
<th>Memory Area</th>
<th>Register description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>0x2000 0000 - 0x2000 8FFF</td>
<td>36 KB</td>
<td>SRAM</td>
<td>Section 2.3 on page 59</td>
</tr>
<tr>
<td></td>
<td>0x2000 9000 - 0x3FFF FFFF</td>
<td>~512 MB</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1FFF 7800 - 0x1FFF 787F</td>
<td>128 B</td>
<td>Option bytes</td>
<td>Section 3.4 on page 72</td>
</tr>
<tr>
<td></td>
<td>0x1FFF 7500 - 0x1FFF 77FF</td>
<td>768 B</td>
<td>Engineering bytes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1FFF 7400 - 0x1FFF 74FF</td>
<td>256 B</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>0x1FFF 7000 - 0x1FFF 73FF</td>
<td>1 KB</td>
<td>OTP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0802 0000 - 0x01FFF D7FF</td>
<td>~384 MB</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0800 0000 - 0x0801 FFFF</td>
<td>128 KB</td>
<td>Main Flash memory</td>
<td>Section 3.3.1 on page 62</td>
</tr>
<tr>
<td></td>
<td>0x0002 0000 - 0x07FF FFFF</td>
<td>~8 MB</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0000 0000 - 0x0001 FFFF</td>
<td>128 KB</td>
<td>Main Flash memory, system memory or SRAM depending on BOOT configuration</td>
<td></td>
</tr>
</tbody>
</table>

### Table 3. STM32G031xx and STM32G041xx memory boundary addresses

<table>
<thead>
<tr>
<th>Type</th>
<th>Boundary address</th>
<th>Size</th>
<th>Memory Area</th>
<th>Register description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>0x2000 0000 - 0x2000 1FFF</td>
<td>8 KB</td>
<td>SRAM</td>
<td>Section 2.3 on page 59</td>
</tr>
<tr>
<td></td>
<td>0x2000 2000 - 0x3FFF FFFF</td>
<td>~512 MB</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1FFF 7800 - 0x1FFF 787F</td>
<td>128 B</td>
<td>Option bytes</td>
<td>Section 3.4 on page 72</td>
</tr>
<tr>
<td></td>
<td>0x1FFF 7500 - 0x1FFF 77FF</td>
<td>768 B</td>
<td>Engineering bytes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1FFF 7400 - 0x1FFF 74FF</td>
<td>256 B</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1FFF 7000 - 0x1FFF 73FF</td>
<td>1 KB</td>
<td>OTP</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>0x1FFF 2000 - 0x1FFF 6FFF</td>
<td>~20 KB</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0801 0000 - 0x01FFF D7FF</td>
<td>~384 MB</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0800 0000 - 0x0800 FFFF</td>
<td>64 KB</td>
<td>Main Flash memory</td>
<td>Section 3.3.1 on page 62</td>
</tr>
<tr>
<td></td>
<td>0x0001 0000 - 0x07FF FFFF</td>
<td>~8 MB</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0000 0000 - 0x0000 FFFF</td>
<td>64 KB</td>
<td>Main Flash memory, system memory or SRAM depending on BOOT configuration</td>
<td></td>
</tr>
</tbody>
</table>
The following table gives the boundary addresses of the peripherals.

### Table 4. STM32G0x1 peripheral register boundary addresses

<table>
<thead>
<tr>
<th>Bus</th>
<th>Boundary address</th>
<th>Size</th>
<th>Peripheral</th>
<th>Peripheral register map</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0xE000 0000 - 0xE00F FFFF</td>
<td>1MB</td>
<td>Cortex®-M0+ internal peripherals</td>
<td>-</td>
</tr>
<tr>
<td>IOPORT</td>
<td>0x5000 1800 - 0x5FFF FFFF</td>
<td>~256 MB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5000 1400 - 0x5000 17FF</td>
<td>1 KB</td>
<td>GPIOF</td>
<td>Section 6.4.12 on page 211</td>
</tr>
<tr>
<td></td>
<td>0x5000 1000 - 0x5000 13FF</td>
<td>1 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5000 0C00 - 0x5000 0FFF</td>
<td>1 KB</td>
<td>GPIOD</td>
<td>Section 6.4.12 on page 211</td>
</tr>
<tr>
<td></td>
<td>0x5000 0800 - 0x5000 0BFF</td>
<td>1 KB</td>
<td>GPIOC</td>
<td>Section 6.4.12 on page 211</td>
</tr>
<tr>
<td></td>
<td>0x5000 0400 - 0x5000 07FF</td>
<td>1 KB</td>
<td>GPIOB</td>
<td>Section 6.4.12 on page 211</td>
</tr>
<tr>
<td></td>
<td>0x5000 0000 - 0x5000 03FF</td>
<td>1 KB</td>
<td>GPIOA</td>
<td>Section 6.4.12 on page 211</td>
</tr>
<tr>
<td>AHB</td>
<td>0x4002 6400 - 0x4002 FFFF</td>
<td>~256 MB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4002 6000 - 0x4002 63FF</td>
<td>1 KB</td>
<td>AES</td>
<td>Section 19.7.18 on page 482</td>
</tr>
<tr>
<td></td>
<td>0x4002 5400 - 0x4002 5FF</td>
<td>3 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4002 5000 - 0x4002 53FF</td>
<td>1 KB</td>
<td>RNG</td>
<td>Section 18.7.4 on page 428</td>
</tr>
<tr>
<td></td>
<td>0x4002 3400 - 0x4002 3FF</td>
<td>3 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4002 3000 - 0x4002 33FF</td>
<td>1 KB</td>
<td>CRC</td>
<td>Section 13.4.6 on page 301</td>
</tr>
<tr>
<td></td>
<td>0x4002 2400 - 0x4002 2FF</td>
<td>3 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4002 2000 - 0x4002 23FF</td>
<td>1 KB</td>
<td>FLASH</td>
<td>Section 3.7.15 on page 100</td>
</tr>
<tr>
<td></td>
<td>0x4002 1C00 - 0x4002 1FF</td>
<td>3 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4002 1800 - 0x4002 1BFF</td>
<td>1 KB</td>
<td>EXTI</td>
<td>Section 12.5.11 on page 294</td>
</tr>
<tr>
<td></td>
<td>0x4002 1400 - 0x4002 17FF</td>
<td>1 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4002 1000 - 0x4002 13FF</td>
<td>1 KB</td>
<td>RCC</td>
<td>Section 5.4.23 on page 192</td>
</tr>
<tr>
<td></td>
<td>0x4002 0C00 - 0x4002 0FF</td>
<td>1 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4002 0800 - 0x4002 0BFF</td>
<td>2 KB</td>
<td>DMAMUX</td>
<td>Section 10.6.7 on page 276</td>
</tr>
<tr>
<td></td>
<td>0x4002 0400 - 0x4002 07FF</td>
<td>1 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4002 0000 - 0x4002 03FF</td>
<td>1 KB</td>
<td>DMA</td>
<td>Section 9.6.7 on page 259</td>
</tr>
<tr>
<td>APB</td>
<td>0x4001 5C00 - 0x4001 FFFF</td>
<td>32 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4001 5800 - 0x4001 5BFF</td>
<td>1 KB</td>
<td>DBG</td>
<td>Section 37.10.5 on page 1224</td>
</tr>
<tr>
<td></td>
<td>0x4001 4C00 - 0x4001 57FF</td>
<td>3 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4001 4800 - 0x4001 4BFF</td>
<td>1 KB</td>
<td>TIM17</td>
<td>Section 24.6.22 on page 786</td>
</tr>
<tr>
<td></td>
<td>0x4001 4400 - 0x4001 47FF</td>
<td>1 KB</td>
<td>TIM16</td>
<td>Section 24.6.22 on page 786</td>
</tr>
</tbody>
</table>
### Table 4. STM32G0x1 peripheral register boundary addresses (continued)

<table>
<thead>
<tr>
<th>Bus</th>
<th>Boundary address</th>
<th>Size</th>
<th>Peripheral</th>
<th>Peripheral register map</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0x4001 4000 - 0x4001 43FF</td>
<td>1 KB</td>
<td>TIM15</td>
<td>Section 24.6.22 on page 786</td>
</tr>
<tr>
<td></td>
<td>0x4001 3C00 - 0x4001 3FF</td>
<td>1 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4001 3800 - 0x4001 3BF</td>
<td>1 KB</td>
<td>USART1</td>
<td>Section 32.7.15 on page 1037</td>
</tr>
<tr>
<td></td>
<td>0x4001 3400 - 0x4001 37FF</td>
<td>1 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4001 3000 - 0x4001 33FF</td>
<td>1 KB</td>
<td>SPI1/I2S1</td>
<td>Section 34.9.10 on page 1150</td>
</tr>
<tr>
<td></td>
<td>0x4001 2C00 - 0x4001 2FF</td>
<td>1 KB</td>
<td>TIM1</td>
<td>Section 20.4 on page 546</td>
</tr>
<tr>
<td></td>
<td>0x4001 2B00 - 0x4001 2FF</td>
<td>1 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4001 2400 - 0x4001 27FF</td>
<td>1 KB</td>
<td>ADC</td>
<td>Section 14.12.17 on page 361</td>
</tr>
<tr>
<td></td>
<td>0x4001 0400 - 0x4001 23FF</td>
<td>8 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4001 0200 - 0x4001 03FF</td>
<td>1 KB</td>
<td>COMP</td>
<td>Section 17.6.3 on page 415</td>
</tr>
<tr>
<td></td>
<td>0x4001 0080 - 0x4001 01FF</td>
<td>1 KB</td>
<td>SYSCFG(ITLINE)</td>
<td>Section 7.1.35 on page 231</td>
</tr>
<tr>
<td></td>
<td>0x4001 0030 - 0x4001 007F</td>
<td>1 KB</td>
<td>VREFBUF</td>
<td>Section 16.3.3 on page 403</td>
</tr>
<tr>
<td></td>
<td>0x4001 0000 - 0x4001 002F</td>
<td>1 KB</td>
<td>SYSCFG</td>
<td>Section 7.1.35 on page 231</td>
</tr>
<tr>
<td></td>
<td>0x4000 B400- 0x4000 FFFF</td>
<td>19 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4000 B000 - 0x4000 B3FF</td>
<td>1 KB</td>
<td>TAMP (+ BKP registers)</td>
<td>Section 30.6.9 on page 882</td>
</tr>
<tr>
<td></td>
<td>0x4000 A800 - 0x4000 AFFF</td>
<td>2 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4000 A400 - 0x4000 A7FF</td>
<td>1 KB</td>
<td>UCPD2</td>
<td>Section 35.7.15 on page 1190</td>
</tr>
<tr>
<td></td>
<td>0x4000 A000 - 0x4000 A3FF</td>
<td>1 KB</td>
<td>UCPD1</td>
<td>Section 35.7.15 on page 1190</td>
</tr>
<tr>
<td></td>
<td>0x4000 9800 - 0x4000 9FFF</td>
<td>2 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4000 9400 - 0x4000 97FF</td>
<td>1 KB</td>
<td>LPTIM2</td>
<td>Section 25.7.10 on page 811</td>
</tr>
<tr>
<td></td>
<td>0x4000 8400 - 0x4000 93FF</td>
<td>4 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4000 8000 - 0x4000 83FF</td>
<td>1 KB</td>
<td>LPUART1</td>
<td>Section 32.7.15 on page 1037</td>
</tr>
<tr>
<td></td>
<td>0x4000 7C00 - 0x4000 7FFF</td>
<td>1 KB</td>
<td>LPTIM1</td>
<td>Section 25.7.10 on page 811</td>
</tr>
<tr>
<td></td>
<td>0x4000 7800 - 0x4000 7BFF</td>
<td>1 KB</td>
<td>HDMI-CEC</td>
<td>Section 36.7.7 on page 1211</td>
</tr>
<tr>
<td></td>
<td>0x4000 7400 - 0x4000 7FFF</td>
<td>1 KB</td>
<td>DAC</td>
<td>Section 15.7.21 on page 399</td>
</tr>
<tr>
<td></td>
<td>0x4000 7000 - 0x4000 73FF</td>
<td>1 KB</td>
<td>PWR</td>
<td>Section 4.4.18 on page 140</td>
</tr>
<tr>
<td></td>
<td>0x4000 5C00 - 0x4000 6FF</td>
<td>5 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4000 5800 - 0x4000 5BFF</td>
<td>1 KB</td>
<td>I2C2</td>
<td>Section 31.7.12 on page 949</td>
</tr>
<tr>
<td></td>
<td>0x4000 5400 - 0x4000 57FF</td>
<td>1 KB</td>
<td>I2C1</td>
<td>Section 31.7.12 on page 949</td>
</tr>
<tr>
<td></td>
<td>0x4000 5000 - 0x4000 53FF</td>
<td>1 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4000 4C00 - 0x4000 4FFF</td>
<td>1 KB</td>
<td>USART4</td>
<td>Section 32.7.15 on page 1037</td>
</tr>
<tr>
<td></td>
<td>0x4000 4800 - 0x4000 4BFF</td>
<td>1 KB</td>
<td>USART3</td>
<td>Section 32.7.15 on page 1037</td>
</tr>
<tr>
<td></td>
<td>0x4000 4400 - 0x4000 47FF</td>
<td>1 KB</td>
<td>USART2</td>
<td>Section 32.7.15 on page 1037</td>
</tr>
<tr>
<td></td>
<td>0x4000 3C00 - 0x4000 43FF</td>
<td>2 KB</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4000 3800 - 0x4000 3BFF</td>
<td>1 KB</td>
<td>SPI2</td>
<td>Section 34.9.10 on page 1150</td>
</tr>
</tbody>
</table>
2.3 Embedded SRAM

STM32G071xx and STM32G081xx devices feature 32 Kbytes of SRAM (static RAM) with parity check enabled and 36 Kbytes with parity check disabled. STM32G031xx and STM32G041xx devices have 8 Kbytes of SRAM regardless of the parity check configuration.

The SRAM can be accessed by bytes, half-words (16 bits) or full words (32 bits), at maximum system clock frequency without wait state and thus by both CPU and DMA.

Parity check

The user can enable the parity check using the option bit RAM_PARITY_CHECK in the user option byte (refer to Section 3.4: FLASH option bytes).

The data bus width is 36 bits because 4 bits are available for parity check (1 bit per byte) in order to increase memory robustness, as required for instance by Class B or SIL norms.

The parity bits are computed and stored when writing into the SRAM. Then, they are automatically checked when reading. If one bit fails, an NMI is generated. The same error can also be linked to the BRK_IN Break input of TIM1/15/16/17, with the SRAM_PARITY_LOCK control bit in the SYSCFG configuration register 2 (SYSCFG_CFRG2). The SRAM Parity Error flag (SRAM_PEF) is available in the SYSCFG configuration register 2 (SYSCFG_CFRG2).

Note: When enabling the SRAM parity check, it is advised to initialize by software the whole SRAM at the beginning of the code, to avoid getting parity errors when reading non-initialized locations.
2.4 Flash memory overview

The Flash memory is composed of two distinct physical areas:

- The main Flash memory block. It contains the application program and user data if necessary.
- The information block. It is composed of three parts:
  - Option bytes for hardware and memory protection user configuration.
  - System memory which contains the proprietary boot loader code.
  - OTP (one-time programmable) area

Please refer to Section 3: Embedded Flash memory (FLASH) for more details.

The Flash interface implements instruction access and data access based on the AHB protocol. It implements the prefetch buffer that speeds up CPU code execution. It also implements the logic necessary to carry out the Flash memory operations (Program/Erase) controlled through the Flash registers.

2.5 Boot configuration

In the STM32G0x1, three different boot modes can be selected through the BOOT0 pin, BOOT_LOCK bit in FLASH_SECR register, and boot configuration bits nBOOT1, BOOT_SEL and nBOOT0 in the User option byte, as shown in the following table.

<table>
<thead>
<tr>
<th>Boot mode configuration</th>
<th>Selected boot area</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOT_LOCK bit</td>
<td>nBOOT1 bit</td>
</tr>
<tr>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
</tr>
</tbody>
</table>

The boot mode configuration is latched on the 4th rising edge of SYSCLK after a reset. It is up to the user to set boot mode configuration related to the required boot mode.

The boot mode configuration is also re-sampled when exiting from Standby mode. Consequently they must be kept in the required Boot mode configuration in Standby mode. After this startup delay has elapsed, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code execution from the boot memory at 0x0000 0004.

Depending on the selected boot mode, main Flash memory, system memory or SRAM is accessible as follows:

- Boot from main Flash memory: the main Flash memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space
(0x0800 0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x0800 0000.

- Boot from system memory: the system memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space 0x1FF0000.
- Boot from the embedded SRAM: the SRAM is aliased in the boot memory space (0x0000 0000), but it is still accessible from its original memory space (0x2000 0000).

**Forcing boot from user Flash memory**

The BOOT_LOCK bit allows forcing a unique entry point in the main Flash memory for boot, regardless of the other boot mode configuration bits. See section *Forcing boot from Flash memory*.

**Empty check**

Internal empty check flag (the EMPTY bit of the FLASH access control register (FLASH_ACR)) is implemented to allow easy programming of virgin devices by the boot loader. This flag is used when BOOT0 pin is defining Main Flash memory as the target boot area. When the flag is set, the device is considered as empty and System memory (boot loader) is selected instead of the Main Flash as a boot area to allow user to program the Flash memory.

This flag is updated only during Option bytes loading: it is set when the content of the address 0x08000 0000 is read as 0xFFFF FFFF, otherwise it is cleared. It means a power reset or setting of OBL_LAUNCH bit in FLASH_CR register is needed to clear this flag after programming of a virgin device to execute user code after System reset. The EMPTY bit can also directly be written by software.

*Note: If the device is programmed for a first time but the Option bytes are not reloaded, the device will still select System memory as a boot area after a System reset.*

**Physical remap**

Once the boot mode is selected, the application software can modify the memory accessible in the code area. This modification is performed by programming the MEM_MODE bits in the SYSCFG configuration register 1 (SYSCFG_CFGR1).

**Embedded boot loader**

The embedded boot loader is located in the System memory, programmed by ST during production. It is used to reprogram the Flash memory using one of the following serial interfaces:

- USART on pins PA2/PA3, PA9/PA10, or (on STM32G071xx and STM32G081xx only) PC10/PC11
- I2C on pins PB6/PB7 or PB10/PB11
- on STM32G071xx and STM32G081xx only, SPI on pins PA4/PA5, PA6/PA7, PB12/PB13 or PB14/PB15

For further details, refer to AN2606.
3  Embedded Flash memory (FLASH)

3.1  FLASH Introduction

The Flash memory interface manages CPU (Cortex®-M0+) AHB to the Flash memory. It implements erase and program Flash memory operations, read and write protection, and security mechanisms.

The Flash memory interface accelerates code execution with a system of instruction prefetch and cache lines.

3.2  FLASH main features

- Up to 128 Kbytes of Flash memory single bank architecture
- Memory organization: 1 bank
  - Main memory: up to 128 Kbytes
  - Page size: 2 Kbytes
  - Subpage size: 512 bytes
- 72-bit wide data read (64 bits plus 8 ECC bits)
- 72-bit wide data write (64 bits plus 8 ECC bits)
- Page erase (2 Kbytes), and mass erase

Flash memory interface features:
- Flash memory read operations
- Flash memory program/erase operations
- Read protection activated by option (RDP)
- Two write protection areas selected by option (WRP)
- Two proprietary code read protection areas selected by option (PCROP)
- Secureable memory area
- Flash memory empty check
- Prefetch buffer
- CPU instruction cache: two cache lines of 64 bits (16 bytes RAM)
- Error code correction (ECC): eight bits for 64 bits
- Option byte loader

3.3  FLASH functional description

3.3.1  FLASH memory organization

The Flash memory is organized as 72-bit-wide memory cells (64 bits plus 8 ECC bits) that can be used for storing both code and data constants.
The Flash memory is organized as follows:

- A Main memory block containing 64 pages of 2 Kbytes, each page with eight rows of 256 bytes.
- An Information block containing:
  - System memory from which the CPU boots in System memory boot mode. The area is reserved and contains the boot loader used to reprogram the Flash memory through one of the following interfaces: USART1, USART2, I2C1, I2C2, and on STM32G071xx and STM32G081xx devices also through USART3, SPI1, and SPI2. On the manufacturing line, the devices are programmed and protected against spurious write/erase operations. For further details, refer to the AN2606 available from www.st.com.
  - 1 Kbyte (128 double words) OTP (one-time programmable) for user data. The OTP data cannot be erased and can be written only once. If only one bit is at 0, the entire double word (64 bits) cannot be written anymore, even with the value 0x0000 0000 0000 0000. The OTP area cannot be read when RDP level is 1 and boot source is not the Main Flash memory area.
  - Option bytes for user configuration.

The Flash memory in a single bank is mapped into Main memory area and an information block, as shown in the following table Table 6.

<table>
<thead>
<tr>
<th>Area</th>
<th>Addresses</th>
<th>Size (bytes)</th>
<th>STM32G071xx and STM32G081xx</th>
<th>STM32G031xx and STM32G041xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main memory</td>
<td>0x0800 0000 - 0x0800 07FF</td>
<td>2 K</td>
<td>Page 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0800 0800 - 0x0800 0FFF</td>
<td>2 K</td>
<td>Page 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0800 1000 - 0x0800 17FF</td>
<td>2 K</td>
<td>Page 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0800 1800 - 0x0800 1FFF</td>
<td>2 K</td>
<td>Page 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0800 F800 - 0x0800 FFFF</td>
<td>2 K</td>
<td>Page 31</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>0x0801 0000 - 0x0801 FFFF</td>
<td>2 K</td>
<td>Page 32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0801 F000 - 0x0801 7FFF</td>
<td>2 K</td>
<td>Page 62</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0801 F800 - 0x0801 FFFF</td>
<td>2 K</td>
<td>Page 63</td>
<td></td>
</tr>
<tr>
<td>Information block</td>
<td>0x1FFF 0000 - 0x1FFF 6FF</td>
<td>28 K(1)</td>
<td>System memory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1FFF 7000 - 0x1FFF 73FF</td>
<td>1 K</td>
<td>OTP area</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1FFF 7500 - 0x1FFF 77FF</td>
<td>768</td>
<td>Engineering bytes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1FFF 7800 - 0x1FFF 787F</td>
<td>128</td>
<td>Option bytes</td>
<td></td>
</tr>
</tbody>
</table>

1. 8 K on STM32G031xx and STM32G041xx devices, from 0x1FFF 0000 to 0x1FFF 1FFF
3.3.2 FLASH empty check

During the OBL phase, after loading all options, the Flash memory interface checks whether the first location of the Main memory is programmed. The result of this check in conjunction with the boot0 and boot1 information is used to determine where the system has to boot from. It prevents the system to boot from Main Flash memory area when i.e. no user code has been programmed.

The Main Flash memory empty check status can be read from the EMPTY bit in FLASH access control register (FLASH_ACR). Software can modify the Main Flash memory empty status by writing an appropriate value to the EMPTY bit.

3.3.3 FLASH error code correction (ECC)

Data in Flash memory words are 72-bits wide: eight bits are added per each double word (64 bits). The ECC mechanism supports:

- One error detection and correction
- Two errors detection

When one error is detected and corrected, the flag ECCC (ECC correction) is set in FLASH ECC register (FLASH_ECCR). If ECCCIE is set, an interrupt is generated.

When two errors are detected, a flag ECCD (ECC detection) is set in FLASH ECC register (FLASH_ECCR). In this case, a NMI is generated.

When an ECC error is detected, the address of the failing double word is saved in ADDR_ECC[16:0] bitfield of the FLASH_ECCR register. ADDR_ECC[2:0] are always cleared. The bus-ID of the CPU accessing the address is saved in CPUID[2:0].

While ECCC or ECCD is set, FLASH_ECCR is not updated if a new ECC error occurs. FLASH_ECCR is updated only when ECC flags are cleared.

Note: For a virgin data: 0xFF FFFF FFFF FFFF FFFF, one error is detected and corrected, but two errors detection is not supported.

When an ECC error is reported, a new read at the failing address may not generate an ECC error if the data is still present in the current buffer, even if ECCC and ECCD are cleared. If this is not the desired behavior, the user must reset the cache.

3.3.4 FLASH read access latency

To correctly read data from Flash memory, the number of wait states (LATENCY) must be correctly programmed in the FLASH access control register (FLASH_ACR) according to the frequency of the Flash (HCLK) memory clock and the internal voltage range of the device V\textsubscript{CORE}. Refer to Section 4.1.4: Dynamic voltage scaling management. Table 7 shows the correspondence between wait states and Flash memory clock frequency.

<table>
<thead>
<tr>
<th>Wait states (WS) (LATENCY)</th>
<th>HCLK (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>V\textsubscript{CORE} Range 1</td>
</tr>
<tr>
<td>0 WS (1 HCLK cycles)</td>
<td>\leq 24</td>
</tr>
<tr>
<td>1 WS (2 HCLK cycles)</td>
<td>\leq 48</td>
</tr>
<tr>
<td>2 WS (3 HCLK cycles)</td>
<td>\leq 64</td>
</tr>
</tbody>
</table>
After power reset, the HCLK clock frequency is 16 MHz in Range 1 and 0 wait state (WS) is configured in the FLASH_ACR register.

When wakeup from Standby, the HCLK clock frequency is 16 MHz in Range 1 and 0 wait state (WS) is configured in the FLASH_ACR register.

When changing the Flash memory clock frequency or Range, the following software sequences must be applied in order to tune the number of wait states needed to access the Flash memory:

**Increasing the CPU frequency**

1. Program the new number of wait states to the LATENCY bits of the *FLASH access control register (FLASH_ACR)*.
2. Check that the new number of wait states is taken into account to access the Flash memory by reading back the LATENCY bits of the *FLASH access control register (FLASH_ACR)*, and wait until the programmed new number is read.
3. Modify the system clock source by writing the SW bits of the RCC_CFGR register.
4. If needed, modify the core clock prescaler by writing the HPRE bits of RCC_CFGR register.
5. Optionally, check that the new system clock source or/and the new core clock prescaler value is/are taken into account by reading the clock source status (SWS bits) of the RCC_CFGR register, or/and the AHB prescaler value (HPREF bit), of the RCC_CFGR register.

**Decreasing the CPU frequency**

1. Modify the system clock source by writing the SW bits of the RCC_CFGR register.
2. If needed, modify the core clock prescaler by writing the HPRE bits of RCC_CFGR.
3. Check that the new system clock source or/and the new core clock prescaler value is/are taken into account by reading the clock source status (SWS bits) of the RCC_CFGR register, or/and the AHB prescaler value (HPREF bit), of the RCC_CFGR register, and wait until the programmed new system clock source or/and new Flash memory clock prescaler value is/are read.
4. Program the new number of wait states to the LATENCY bits of the *FLASH access control register (FLASH_ACR)*.
5. Optionally, check that the new number of wait states is used to access the Flash memory by reading back the LATENCY bits of the *FLASH access control register (FLASH_ACR)*.

**3.3.5 FLASH memory acceleration**

**Instruction prefetch**

Each Flash memory read operation provides 64 bits from either two instructions of 32 bits or four instructions of 16 bits according to the program launched. This 64-bits current instruction line is saved in a current buffer. So, in case of sequential code, at least two CPU cycles are needed to execute the previous read instruction line. Prefetch on the CPU S-bus can be used to read the next sequential instruction line from the Flash memory while the current instruction line is being requested by the CPU.
Prefetch is enabled by setting the PRFTEN bit of the FLASH access control register (FLASH_ACR). This feature is useful if at least one wait state is needed to access the Flash memory.

When the code is not sequential (branch), the instruction may not be present in the currently used instruction line or in the prefetched instruction line. In this case (miss), the penalty in terms of number of cycles is at least equal to the number of wait states.

If a loop is present in the current buffer, no new access is performed.

**Cache memory**

To limit the time lost due to jumps, it is possible to retain two cache lines of 64 bits (16 bytes) in the instruction cache memory. This feature can be enabled by setting the instruction cache enable (ICEN) bit of the FLASH access control register (FLASH_ACR). Each time a miss occurs (requested data not present in the currently used instruction line, in the prefetched instruction line or in the instruction cache memory), the line read is copied into the instruction cache memory. If some data contained in the instruction cache memory are requested by the CPU, they are provided without inserting any delay. Once all the instruction cache memory lines are filled, the LRU (least recently used) policy is used to determine the line to replace in the instruction memory cache. This feature is particularly useful in case of code containing loops.

The Instruction cache memory is enabled after system reset.

CPU literal pools are fetched from Flash memory through the S-bus during the execution stage of the CPU pipeline. Each S-bus read access fetches 64 bits saved in a current buffer. The CPU pipeline is consequently stalled until the requested literal pool is provided.

No data cache is available on Cortex®-M0+.

### 3.3.6 FLASH program and erase operations

The STM32G071xx and STM32G081xx embedded Flash memory can be programmed using in-circuit programming or in-application programming.

The in-circuit programming (ICP) method is used to update the entire contents of the Flash memory, using SWD protocol or the supported interfaces by the system boot loader, to load the user application for the CPU, into the microcontroller. ICP offers quick and efficient design iterations and eliminates unnecessary package handling or socketing of devices.

In contrast to the ICP method, in-application programming (IAP) can use any communication interface supported by the microcontroller (I/Os, UART, I²C, SPI, etc.) to download programming data into memory. IAP allows the user to re-program the Flash memory while the application is running. Nevertheless, part of the application has to have been previously programmed in the Flash memory using ICP.

The contents of the Flash memory are not guaranteed if a device reset occurs during a Flash memory operation.

During a program/erase operation to the Flash memory, any attempt to read the Flash memory will stall the bus. The read operation will proceed correctly once the program/erase operation has completed.
Unlocking the Flash memory

After reset, write into the Flash control register (FLASH_CR) is not allowed so as to protect the Flash memory against possible unwanted operations due, for example, to electric disturbances. The following sequence unlocks these registers:
1. Write KEY1 = 0x4567 0123 in the Flash key register (FLASH_KEYR)
2. Write KEY2 = 0xCDEF 89AB in the Flash key register (FLASH_KEYR).

Any wrong sequence will lock the FLASH_CR registers until the next system reset. In the case of a wrong key sequence, a bus error is detected and a Hard Fault interrupt is generated.

The Flash CR registers can be locked again by software by setting the LOCK bit in one of these registers.

**Note:** The Flash CR register cannot be written when the BSY1 bit of the Flash status register (FLASH_SR) is set. Any attempt to write to this register with the BSY1 bit set causes the AHB bus to stall until the BSY1 bit is cleared.

### 3.3.7 Flash Main memory erase sequences

The Flash memory erase operation can be performed at page level (page erase), or on the whole memory (mass erase). Mass erase does not affect the Information block (system Flash memory, OTP and option bytes).

**Flash memory page erase**

When a page is protected by PCROP or WRP, it is not erased and the WRPERR bit is set.

<table>
<thead>
<tr>
<th>SEC_PROT</th>
<th>PCROP</th>
<th>WRP</th>
<th>PCROP_RDP</th>
<th>Comment</th>
<th>WRPERR</th>
<th>CPU bus error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No</td>
<td>No</td>
<td>x</td>
<td>Page is erased</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>Yes</td>
<td></td>
<td>Page erase aborted</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
<td></td>
<td>(no page erase started)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td></td>
<td></td>
<td>Protected pages only</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

To erase a page (2 Kbytes), follow the procedure below:
1. Check that no Flash memory operation is ongoing by checking the BSY1 bit of the Flash status register (FLASH_SR).
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the PER bit and select the page to erase (PNB) in the Flash control register (FLASH_CR).
4. Set the STRT bit of the Flash control register (FLASH_CR).
5. Wait until the BSY1 bit of the Flash status register (FLASH_SR) is cleared.

**Note:** The internal oscillator HSI16 (16 MHz) is enabled automatically when STRT bit is set, and disabled automatically when STRT bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.
Flash memory mass erase

When PCROP or WRP is enabled the Flash memory mass erase is aborted, no erase starts, and the WRPERR bit is set.

<table>
<thead>
<tr>
<th>SEC_PROT</th>
<th>PCROP</th>
<th>WRP</th>
<th>PCROP_RDP</th>
<th>Comment</th>
<th>WRPERR</th>
<th>CPU bus error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No</td>
<td>No</td>
<td>x</td>
<td>Memory is erased</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>Yes</td>
<td></td>
<td>Erase aborted (no erase started)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td></td>
<td></td>
<td>Erase aborted (no erase started)</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

To perform a mass erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY1 bit of the FLASH status register (FLASH_SR).
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the MER1 bit of the FLASH control register (FLASH_CR).
4. Set the STRT bit of the FLASH control register (FLASH_CR).
5. Wait until the BSY1 bit of the FLASH status register (FLASH_SR) is cleared.

Note: The internal oscillator HSI16 (16 MHz) is enabled automatically when STRT bit is set, and disabled automatically when STRT bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.

3.3.8 FLASH Main memory programming sequences

The Flash memory is programmed 72 bits (64-bit data plus 8-bit ECC) at a time. Programming in a previously programmed address is not allowed, any attempt will set PROGERR flag of the FLASH status register (FLASH_SR).

It is only possible to program a double word (2 x 32-bit data).

- Any attempt to write byte (8 bits) or half-word (16 bits) will set SIZERR flag of the FLASH status register (FLASH_SR).
- Any attempt to write a double word that is not aligned with a double word address will set PGAERR flag of the FLASH status register (FLASH_SR).

Standard programming

The Flash memory programming sequence in standard mode is as follows:
1. Check that no Main Flash memory operation is ongoing by checking the BSY1 bit of the
FLASH status register (FLASH_SR).
2. Check and clear all error programming flags due to a previous programming. If not,
PGSERR is set.
3. Set the PG bit of the FLASH control register (FLASH_CR).
4. Perform the data write operation at the desired memory address, inside Main memory
block or OTP area. Only double word (64 bits) can be programmed.
   a) Write a first word in an address aligned with double word
   b) Write the second word.
5. Wait until the BSY1 bit of the FLASH status register (FLASH_SR) is cleared.
6. Check that EOP flag of the FLASH status register (FLASH_SR) is set (programming
operation succeeded), and clear it by software.
7. Clear the PG bit of the FLASH control register (FLASH_CR) if there no more
programming request anymore.

**Note:** When the Flash memory interface has received a good sequence (a double word),
programming is automatically launched and BSY1 bit is set. The internal oscillator HSI16
(16 MHz) is enabled automatically when PG bit is set, and disabled automatically when PG
bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.
If the user needs to program only one word, double word must be completed with the erase
value 0xFFFF FFFF to launch automatically the programming.
ECC is calculated from the double word to program.

**Fast programming**
The main purpose of this mode is to reduce the page programming time. It is achieved by
eliminating the need for verifying the Flash memory locations before they are programmed,
thus saving the time of high voltage ramping and falling for each double word.

This mode allows programming a row (32 double words = 256 bytes).

During fast programming, the Flash memory clock (HCLK) frequency must be at least 8
MHz.

Only the Main memory can be programmed in Fast programming mode.

The Main Flash memory programming sequence in standard mode is described below:
1. Perform a mass or page erase. If not, PGSERR is set.
2. Check that no Main Flash memory operation is ongoing by checking the BSY1 bit of the
FLASH status register (FLASH_SR).
3. Check and clear all error programming flag due to a previous programming.
4. Set the FSTPG bit in FLASH control register (FLASH_CR).
5. Write 32 double words to program a row (256 bytes).
6. Wait until the BSY1 bit of the FLASH status register (FLASH_SR) is cleared.
7. Check that EOP flag of the FLASH status register (FLASH_SR) is set (programming
operation succeeded), and clear it by software.
8. Clear the FSTPG bit of the FLASH status register (FLASH_SR) if there are no more
programming requests anymore.
Note: When attempting to write in Fast programming mode while a read operation is on going, the programming is aborted without any system notification (no error flag is set).

When the Flash memory interface has received the first double word, programming is automatically launched. The BSY1 bit is set when the high voltage is applied for the first double word, and it is cleared when the last double word has been programmed or in case of error. The internal oscillator HSI16 (16 MHz) is enabled automatically when FSTPG bit is set, and disabled automatically when FSTPG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.

The 32 double words must be written successively. The high voltage is kept on the Flash memory for all the programming. Maximum time between two double words write requests is the time programming (around 20 µs). If a second double word arrives after this time programming, fast programming is interrupted and MISSERR is set.

High voltage must not exceed 8 ms for a full row between two erases. This is guaranteed by the sequence of 32 double words successively written with a clock system greater or equal to 8 MHz. An internal time-out counter counts 7 ms when Fast programming is set and stops the programming when time-out is over. In this case the FASTERR bit is set.

If an error occurs, high voltage is stopped and next double word to programmed is not programmed. Anyway, all previous double words have been properly programmed.

Programming errors

Several kind of errors can be detected. In case of error, the Flash memory operation (programming or erasing) is aborted.

- **PROGERR**: Programming Error
  In standard programming: PROGERR is set if the word to write is not previously erased (except if the value to program is full zero).

- **SIZERR**: Size Programming Error
  In standard programming or in fast programming: only double word can be programmed, and only 32-bit data can be written. SIZERR is set if a byte or an half-word is written.

- **PGAERR**: Alignment Programming error
  PGAERR is set if one of the following conditions occurs:
  - In standard programming: the first word to be programmed is not aligned with a double word address, or the second word doesn’t belong to the same double word address.
  - In fast programming: the data to program doesn’t belong to the same row than the previous programmed double words, or the address to program is not greater than the previous one.

- **PGSERR**: Programming Sequence Error
  PGSERR is set if one of the following conditions occurs:
  - In the standard programming sequence or the fast programming sequence: a data is written when PG and FSTPG are cleared.
  - In the standard programming sequence or the fast programming sequence: MER1 and PER are not cleared when PG or FSTPG is set.
  - In the fast programming sequence: the Mass erase is not performed before setting the FSTPG bit.
  - In the mass erase sequence: PG, FSTPG, and PER are not cleared when MER1 is set.
– In the page erase sequence: PG, FSTPG and MER1 are not cleared when PER is set.
– PGSERR is set also if PROGERR, SIZERR, PGAERR, WRPERR, MISSERR, FASTERR or PGSERR is set due to a previous programming error.

- **WRPERR**: Write Protection Error
  WRPERR is set if one of the following conditions occurs:
  – Attempt to program or erase in a write protected area (WRP) or in a PCROP area.
  – Attempt to perform a mass erase when one page or more is protected by WRP or PCROP.
  – The debug features are connected or the boot is executed from SRAM or from system Flash memory when the read protection (RDP) is set to Level 1.
  – Attempt to modify the option bytes when the read protection (RDP) is set to Level 2.

- **MISSERR**: Fast Programming Data Miss Error
  In fast programming: all the data must be written successively. MISSERR is set if the previous data programmation is finished and the next data to program is not written yet.

- **FASTERR**: Fast Programming Error
  In fast programming: FASTERR is set if one of the following conditions occurs:
  – when FSTPG bit is set for more than 8 ms, which generates a time-out detection
  – when the row fast programming has been interrupted by a MISSERR, PGAERR, WRPERR or SIZERR

If an error occurs during a program or erase operation, one of the following error flags of the *FLASH status register (FLASH_SR)* is set:

- **PROGERR, SIZERR, PGAERR, PGSERR, MISSERR** (program error flags)
- **WRPERR** (protection error flag)

In this case, if the error interrupt enable bit ERRIE of the *FLASH control register (FLASH_CR)* is set, an interrupt is generated and the operation error flag OPERR of the *FLASH status register (FLASH_SR)* is set.

*Note:* If several successive errors are detected (for example, in case of DMA transfer to the Flash memory), the error flags cannot be cleared until the end of the successive write request.

### Programming and cache

If an erase operation in Flash memory also concerns data in the instruction cache, the user has to ensure that these data are rewritten before they are accessed during code execution.

*Note:* The cache should be flushed only when it is disabled (ICEN = 0).
3.4  FLASH option bytes

3.4.1  FLASH option byte description

The option bytes are configured by the end user depending on the application requirements. As a configuration example, the watchdog may be selected in hardware or software mode (refer to Section 3.4.2: FLASH option byte programming).

A double word is split up in option bytes as indicated in Table 10.

### Table 10. Option byte format

<table>
<thead>
<tr>
<th>63-56</th>
<th>55-48</th>
<th>47-40</th>
<th>39-32</th>
<th>31-24</th>
<th>23-16</th>
<th>15-8</th>
<th>7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complemented option byte 3</td>
<td>Complemented option byte 2</td>
<td>Complemented option byte 1</td>
<td>Complemented option byte 0</td>
<td>Option byte 3</td>
<td>Option byte 2</td>
<td>Option byte 1</td>
<td>Option byte 0</td>
</tr>
</tbody>
</table>

The organization of these bytes in the information block is shown in Table 11. The option bytes can be read from the memory locations listed in Table 11 or from the Option byte registers:

- **FLASH option register (FLASH_OPTR)**
- **FLASH PCROP area A start address register (FLASH_PCROP1ASR)**
- **FLASH PCROP area A end address register (FLASH_PCROP1AER)**
- **FLASH PCROP area B start address register (FLASH_PCROP1BSR)**
- **FLASH PCROP area B end address register (FLASH_PCROP1BER)**
- **FLASH WRP area A address register (FLASH_WRP1AR)**
- **FLASH WRP area A address register (FLASH_WRP1BR)**
- **FLASH security register (FLASH_SECR)**

### Table 11. Organization of option bytes

<table>
<thead>
<tr>
<th>Address(1)</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1FFF7800</td>
<td>Reserved</td>
<td>IRHEN</td>
<td>NRST_MODE</td>
<td>nBOOT0</td>
<td>nBOOT1</td>
<td>nBOOT_SEL</td>
<td>Reserved</td>
<td>RAM_PARITY_CHECK</td>
<td>Reserved</td>
<td>WWDG_SW</td>
<td>WWDG_STBY</td>
<td>IMDG_STOP</td>
<td>IMDG_SW</td>
<td>nRST_SHDN</td>
<td>nRST_SDDBY</td>
<td>nRST_pieces</td>
<td>Reserved</td>
<td>Reserved</td>
<td>RDP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1FFF7808</td>
<td>Reserved</td>
<td>PCROP1A_STRT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1FFF7810</td>
<td>PCROP_RDP</td>
<td>Reserved</td>
<td>PCROP1A_END</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1FFF7818</td>
<td>Reserved</td>
<td>WRPTA_END</td>
<td>Reserved</td>
<td>Reserved</td>
<td>WRPTA_STRT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1FFF7820</td>
<td>Reserved</td>
<td>WRPTB_END</td>
<td>Reserved</td>
<td>Reserved</td>
<td>WRPTB_STRT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1FFF7828</td>
<td>Reserved</td>
<td>PCROP1B_STRT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 11. Organization of option bytes (continued)

<table>
<thead>
<tr>
<th>Address</th>
<th>Address(1)</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1FFF7830</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1FFF7838 to 0x1FFF7868</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1FFF7870</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. The upper 32-bits of the double-word address contain the inverted data from the lower 32 bits.

### User and read protection option bytes

Flash memory address: 0x1FFF 7800

ST production value: 0xFFFF FEAA

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
</tr>
<tr>
<td>29</td>
<td>IRHEN: Internal reset holder enable bit</td>
</tr>
<tr>
<td>0: Internal resets are propagated as simple pulse on NRST pin</td>
<td></td>
</tr>
<tr>
<td>1: Internal resets drives NRST pin low until it is seen as low level</td>
<td></td>
</tr>
<tr>
<td>28:27</td>
<td>NRST_MODE[1:0]</td>
</tr>
<tr>
<td>00: Reserved</td>
<td></td>
</tr>
<tr>
<td>01: Reset Input only: a low level on the NRST pin generates system reset, internal RESET not propagated to the NSRT pin</td>
<td></td>
</tr>
<tr>
<td>10: GPIO: standard GPIO pad functionality, only internal RESET possible</td>
<td></td>
</tr>
<tr>
<td>11: Bidirectional reset: NRST pin configured in reset input/output mode (legacy mode)</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>nBOOT0: nBOOT0 option bit</td>
</tr>
<tr>
<td>0: nBOOT0=0</td>
<td></td>
</tr>
<tr>
<td>1: nBOOT0=1</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>nBOOT1: Boot configuration</td>
</tr>
<tr>
<td>Together with the BOOT0 pin or option bit nBOOT0 (depending on nBOOT_SEL option bit configuration), this bit selects boot mode from the Main Flash memory, SRAM or the System memory. Refer to Section 2.5: Boot configuration.</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>nBOOT_SEL</td>
</tr>
<tr>
<td>0: BOOT0 signal is defined by BOOT0 pin value (legacy mode)</td>
<td></td>
</tr>
<tr>
<td>1: BOOT0 signal is defined by nBOOT0 option bit</td>
<td></td>
</tr>
</tbody>
</table>
Bit 23  Reserved

Bit 22 **RAM_PARITY_CHECK**: SRAM parity check control  
0: SRAM parity check enable  
1: SRAM parity check disable

Bits 21:20  Reserved

Bit 19 **WWDG_SW**: Window watchdog selection  
0: Hardware window watchdog  
1: Software window watchdog

Bit 18 **IWDG_STDBY**: Independent watchdog counter freeze in Standby mode  
0: Independent watchdog counter is frozen in Standby mode  
1: Independent watchdog counter is running in Standby mode

Bit 17 **IWDG_STOP**: Independent watchdog counter freeze in Stop mode  
0: Independent watchdog counter is frozen in Stop mode  
1: Independent watchdog counter is running in Stop mode

Bit 16 **IDWG_SW**: Independent watchdog selection  
0: Hardware independent watchdog  
1: Software independent watchdog

Bit 15 **nRSTS_SHDW**  
0: Reset generated when entering the Shutdown mode  
1: No reset generated when entering the Shutdown mode

Bit 14 **nRST_STDBY**  
0: Reset generated when entering the Standby mode  
1: No reset generated when entering the Standby mode

Bit 13 **nRST_STOP**  
0: Reset generated when entering the Stop mode  
1: No reset generated when entering the Stop mode

Bits 12:11 **BORR_LEV[1:0]**: These bits contain the VDD supply level threshold that releases the reset.  
00: BOR rising level 1 with threshold around 2.1 V  
01: BOR rising level 2 with threshold around 2.3 V  
10: BOR rising level 3 with threshold around 2.6 V  
11: BOR rising level 4 with threshold around 2.9 V

Bits 10:9 **BORF_LEV[1:0]**: These bits contain the VDD supply level threshold that activates the reset.  
00: BOR falling level 1 with threshold around 2.0 V  
01: BOR falling level 2 with threshold around 2.2 V  
10: BOR falling level 3 with threshold around 2.5 V  
11: BOR falling level 4 with threshold around 2.8 V

Bit 8 **BOR_EN**: Brown out reset enable  
0: Configurable brown out reset disabled, power-on reset defined by POR/PDR levels  
1: Configurable brown out reset enabled, values of BORR_LEV and BORF_LEV taken into account

Bits 7:0 **RDP[7:0]**: Read protection level  
0xAA: Level 0, read protection not active  
0xCC: Level 2, chip read protection active  
Others: Level 1, memories read protection active
**PCROP1A start address option bytes**

Flash memory address: 0x1FFF 7808

ST production value: 0xFFFF FFFF

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

PCROP1A_STRT[7:0]

Bits 31:8 Reserved

Bits 7:0 PCROP1A_STRT[7:0]: PCROP1A area start offset

PCROP1A_STRT contains the offset of the first PCROP subpage of the PCROP1A area.

**PCROP1A end address option bytes**

Flash memory address: 0x1FFF 7810

ST production value: 0x0000 0000

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

PCROP1A_END[7:0]

Bit 31 PCROP_RDP: PCROP area erase upon RDP level regression

This bit determines whether the PCROP area (and the totality of the PCROP area boundary pages) is erased by the mass erase triggered by the RDP level regression from Level 1 to Level 0:

0: Not erased

1: Erased

The software can only set this bit. It is automatically reset upon mass erase following the RDP regression from Level 1 to Level 0.

Bits 30:8 Reserved

Bits 7:0 PCROP1A_END[7:0]: PCROP1A area end offset

PCROP1A_END contains the address of the last subpage of the PCROP1A area.

**WRP Area A address option bytes**

Flash memory address: 0x1FFF 7818

ST production value: 0x0000 00FF
### Embedded Flash memory (FLASH)

<table>
<thead>
<tr>
<th>Address</th>
<th>Bits 31:22</th>
<th>Bits 21:16</th>
<th>Bits 15:6</th>
<th>Bits 5:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRP1A_END[5:0]</td>
<td>0x1FFF 7820</td>
<td><strong>WRP1A_END[5:0]</strong></td>
<td>0xFFFF FFFF</td>
<td>0x0000 00FF</td>
</tr>
<tr>
<td>WRP1B_END[5:0]</td>
<td>0x1FFF 7828</td>
<td><strong>WRP1B_END[5:0]</strong></td>
<td>0xFFFF FFFF</td>
<td>0x0000 00FF</td>
</tr>
<tr>
<td>PCROP1B_STRT[7:0]</td>
<td>0x0000 00FF</td>
<td><strong>PCROP1B_STRT[7:0]</strong></td>
<td>0xFFFF FFFF</td>
<td>0x0000 00FF</td>
</tr>
</tbody>
</table>

#### WRP Area B address option bytes

Flash memory address: 0x1FFF 7820
ST production value: 0x0000 00FF

<table>
<thead>
<tr>
<th>Address</th>
<th>Bits 31:22</th>
<th>Bits 21:16</th>
<th>Bits 15:6</th>
<th>Bits 5:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRP1A_END[5:0]</td>
<td>0x1FFF 7820</td>
<td><strong>WRP1A_END[5:0]</strong></td>
<td>0xFFFF FFFF</td>
<td>0x0000 00FF</td>
</tr>
<tr>
<td>WRP1B_END[5:0]</td>
<td>0x1FFF 7828</td>
<td><strong>WRP1B_END[5:0]</strong></td>
<td>0xFFFF FFFF</td>
<td>0x0000 00FF</td>
</tr>
<tr>
<td>PCROP1B_STRT[7:0]</td>
<td>0x0000 00FF</td>
<td><strong>PCROP1B_STRT[7:0]</strong></td>
<td>0xFFFF FFFF</td>
<td>0x0000 00FF</td>
</tr>
</tbody>
</table>

#### PCROP1B start address option bytes

Flash memory address: 0x1FFF 7828
ST production value: 0xFFFF FFFF
### Embedded Flash memory (FLASH)

**PCROP1B end address option bytes**

Flash memory address: 0x1FFF 7830  
ST production value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PCROP1B_STRT[7:0]: PCROP1B area start offset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCROP1B_STRT contains the offset of the first PCROP subpage of the PCROP1B area.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Security option bytes**

Flash memory address: 0x1FFF 7870  
ST production value: 0x0000 0000

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

Bits 31:8 Reserved

Bits 7:0 **PCROP1B_END[7:0]:** PCROP1B area end offset  
PCROP1B_END contains the offset of the last PCROP subpage of the PCROP1B area.

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

Bit 16 **BOOT_LOCK:** used to force boot from user area  
0: Boot based on the pad/option bit configuration  
1: Boot forced from Main Flash memory  

**Caution:** If BOOT_LOCK is set in association with RDP Level 1, the debug capabilities of the device are stopped and the reset value of the DBG_SWEN bit of the FLASH_ACR register becomes zero. If DBG_SWEN bit is not set by the application code after reset, there is no way to recover from this situation.

Bits 15:7 Reserved

Bits 6:0 **SEC_SIZE[6:0]:** Securable memory area size  
Contains the number of securable Flash memory pages.
3.4.2 FLASH option byte programming

After reset, the options related bits of the FLASH control register (FLASH_CR) are write-protected. To run any operation on the option bytes page, the option lock bit OPTLOCK of the FLASH control register (FLASH_CR) must be cleared. The following sequence is used to unlock this register:

1. Unlock the FLASH_CR with the LOCK clearing sequence (refer to Unlocking the Flash memory)
2. Write OPTKEY1=0x08192A3B of the FLASH option key register (FLASH_OPTKEYR)
3. Write OPTKEY2=0x4C5D6E7F of the FLASH option key register (FLASH_OPTKEYR)

Any wrong sequence will lock up the Flash memory option registers until the next system reset. In the case of a wrong key sequence, a bus error is detected and a Hard Fault interrupt is generated.

The user options can be protected against unwanted erase/program operations by setting the OPTLOCK bit by software.

Note: If LOCK is set by software, OPTLOCK is automatically set as well.

Modifying user options

The option bytes are programmed differently from a Main memory user address.

To modify the value of user options, follow the procedure below:

1. Clear OPTLOCK option lock bit with the clearing sequence described above
2. Write the desired values in the FLASH option registers.
3. Check that no Flash memory operation is ongoing, by checking the BSY1 bit of the FLASH status register (FLASH_SR).
4. Set the Options Start bit OPTSTRT of the FLASH control register (FLASH_CR).
5. Wait for the BSY1 bit to be cleared.

Note: Any modification of the value of one option is automatically performed by erasing user option byte pages first, and then programming all the option bytes with the values contained in the Flash memory option registers.

The complementary values are automatically computed and written into the complemented option bytes upon setting the OPTSTRT bit.

Option byte loading

After the BSY1 bit is cleared, all new options are updated into the Flash memory, but not applied to the system. A read from the option registers will still return the last loaded option byte values, the new options will have effect on the system only after they are loaded.

Option bytes loading is performed in two cases:

- when OBL_LAUNCH bit of the FLASH control register (FLASH_CR) is set
- after a power reset (BOR reset or exit from Standby/Shutdown modes)

Option byte loader performs a read of the options block and stores the data into internal option registers. These internal registers configure the system and can be read by software. Setting OBL_LAUNCH generates a reset so the option byte loading is performed under system reset.
Each option bit has also its complement in the same double word. During option loading, a verification of the option bit and its complement allows to check the loading has correctly taken place.

During option byte loading, the options are read by double word. ECC on option words is not taken into account during OBL, but only during direct SW read of option area.

If the word and its complement are matching, the option word/byte is copied into the option register.

If the comparison between the word and its complement fails, a status bit OPTVERR is set. Mismatch values are forced into the option registers:
- For USR OPT option, the value of mismatch is 1 for all option bits, except the bits of BORR_LEV and BORF_LEV bitfields that are 00 (lowest threshold), and the BOR_EN bit that is 0 (BOR disabled).
- For WRP option, the value of mismatch is the default value “No protection”.
- For RDP option, the value of mismatch is the default value “Level 1”.
- For PCROP, the value of mismatch is “all memory protected”.

Upon system reset, the option bytes are copied into the following option registers that can be read and written by software:
- FLASH_OPTR
- FLASH_PCROP1xSR (x = A or B)
- FLASH_PCROP1xER (x = A or B)
- FLASH_WRP1xR (x = A or B)
- FLASH_SECR

These registers are also used to modify options. If these registers are not modified by user, they reflect the options states of the system. See Modifying user options for more details.

3.5 FLASH memory protection

The Main Flash memory can be protected against external accesses with the read protection (RDP). The pages can also be protected against unwanted write (WRP) due to loss of program counter context. The write-protection WRP granularity is 2 Kbytes. Apart from the RDP and WRP, the Flash memory can also be protected against read and write by third party (PCROP). The PCROP granularity (subpage size) is 512 bytes.

3.5.1 FLASH read protection (RDP)

The read protection is activated by setting the RDP option byte and then, by applying a system reset to reload the new RDP option byte. The read protection protects the Main Flash memory, the option bytes, the backup registers (TAMP_BKPxR in TAMP) and the SRAM.

Note: If the read protection is set while the debugger is still connected through SWD, apply power reset instead of system reset.

There are three levels of read protection from no protection (Level 0) to maximum protection or no debug (Level 2).

The Flash memory is protected when the RDP option byte and its complement contain the pair of values shown in Table 12.
The System memory area is read-accessible whatever the protection level. It is never accessible for program/erase operation.

**Level 0: no protection**

Read, program and erase operations within the Main Flash memory area are possible. The option bytes and the backup registers are also accessible by all operations.

**Level 1: Read protection**

Level 1 read protection is set when the RDP byte and the RDP complemented byte contain any value combinations other than [0xAA, 0x55] and [0xCC, 0x33]. Level 1 is the default protection level when RDP option byte is erased.

- **User mode:** Code executing in user mode (boot from user Flash memory) can access Main Flash memory, option bytes and backup registers with all operations.
- **Debug, boot from SRAM, and boot from System memory modes:** In debug mode or when code boots from SRAM or System memory, the Main Flash memory and the backup registers (TAMP_BKPxR in TAMP) are totally inaccessible. In these modes, a read or write access to the Flash memory generates a bus error and a Hard Fault interrupt.

**Caution:** In Level 1 with no PCROP areas defined, it is mandatory to set PCROP_RDP bit to 1 (full mass erase when the RDP level is decreased from Level 1 to Level 0). In Level 1 with a PCROP area defined, user code to protect by RDP but not by PCROP must be placed outside pages containing a PCROP-protected subpage.

**Level 2: No debug**

In this level, the protection Level 1 is guaranteed. In addition, the CPU debug port, the boot from RAM (boot RAM mode) and the boot from System memory (boot loader mode) are no more available. In user execution mode (boot FLASH mode), all operations are allowed on the Main Flash memory.

**Note:** The CPU debug port is also disabled under reset.

**Note:** STMicroelectronics is not able to perform analysis on defective parts on which the Level 2 protection has been set.

**Changing the read protection level**

The read protection level can change:

- from Level 0 to Level 1, upon changing the value of the RDP byte to any value except 0xCC
- from Level 0 or Level 1 to Level 2, upon changing the value of the RDP byte to 0xCC
- from Level 1 to Level 0, upon changing the value of the RDP byte to 0xAA

<table>
<thead>
<tr>
<th>RDP byte value</th>
<th>RDP complement byte value</th>
<th>Read protection level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xAA</td>
<td>0x55</td>
<td>Level 0</td>
</tr>
<tr>
<td>Any values except the combinations [0xAA, 0x55] and [0xCC, 0x33]</td>
<td></td>
<td>Level 1 (default)</td>
</tr>
<tr>
<td>0xCC</td>
<td>0x33</td>
<td>Level 2</td>
</tr>
</tbody>
</table>
Once in Level 2, it is no more possible to modify the read protection level.

With the PCROP_RDP bit of the FLASH PCROP area A end address register (FLASH_PCROP1AER) set, the change from Level 1 to Level 0 triggers full mass erase of the Main Flash memory. The backup registers (TAMP_BKPxR) are also erased. The user options except PCROP protection are set to their previous values copied from FLASH_OPTR and FLASH_WRP1xR (x = A or B). PCROP is disabled. The OTP area is not affected by mass erase and remains unchanged.

With the PCROP_RDP bit cleared, a partial mass erase occurs, only erasing Flash memory pages that do not overlap with PCROP area (do not contain any PCROP-protected subpage). The option bytes are re-programmed with their previous values. This is also true for FLASH_PCROP1xSR and FLASH_PCROP1xER registers (x = A or B).

Table 13: Mass erase upon RDP regression from Level 1 to Level 0

<table>
<thead>
<tr>
<th>PCROP area</th>
<th>PCROP_RDP</th>
<th>Mass erase</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>x</td>
<td>Full</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Flash memory and backup register)</td>
</tr>
<tr>
<td>Part of Flash memory</td>
<td>1</td>
<td>Partial</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Flash memory pages not overlapping with PCROP area, and backup registers)</td>
</tr>
<tr>
<td>Full Flash memory</td>
<td>0</td>
<td>None</td>
</tr>
</tbody>
</table>

Note: Mass erase (full or partial) is only triggered by the RDP regression from Level 1 to Level 0. RDP level increase (Level 0 to Level 1, 1 to 2, or 0 to 2) does not cause any mass erase. To validate the protection level change, the option bytes must be reloaded by setting the OBL_LAUNCH bit of the FLASH control register (FLASH_CR).
Two areas of the Flash memory can be protected against unwanted read and/or write by a third party.

3.5.2 FLASH proprietary code readout protection (PCROP)
The protected area is execute-only: it can only be reached by the STM32 CPU, with an instruction code, while all other accesses (DMA, debug and CPU data read, write and erase) are strictly prohibited. The PCROP areas have subpage (512-byte) granularity. An additional option bit (PCROP_RDP) allows to select if the PCROP area is erased or not when the RDP protection is changed from Level 1 to Level 0 (refer to Changing the read protection level).

Each PCROP area is defined by a start subpage offset and an end subpage offset into the Flash memory. These offsets are defined in the PCROP address registers FLASH PCROP area A start address register (FLASH_PCROP1ASR), FLASH PCROP area A end address register (FLASH_PCROP1AER), FLASH PCROP area B start address register (FLASH_PCROP1BSR) and FLASH PCROP area B end address register (FLASH_PCROP1BER).

A PCROP area is defined from the address:

Flash memory Base address + \([\text{PCROP1x\_STRT} \times 0x200]\) (included)

to the address:

Flash memory Base address + \([(\text{PCROP1x\_END} + 1) \times 0x200]\) (excluded).

The minimum PCROP area size is two PCROP subpages (2 x 512 bytes):

\[\text{PCROP1x\_END} = \text{PCROP1x\_STRT} + 1.\]

When

\[\text{PCROP1x\_END} = \text{PCROP1x\_STRT},\]

the full Flash memory is PCROP-protected.

For example, to PCROP-protect the address area from 0x0800 0800 to 0x0800 13FF, set the PCROP start subpage bitfield of the FLASH_PCROP1xSR register and the PCROP end subpage bitfield of the FLASH_PCROP1xER register (x = A or B) as follows:

- \[\text{PCROP1x\_STRT} = 0x04 \quad (\text{PCROP area start address 0x0800 0800})\]
- \[\text{PCROP1x\_END} = 0x09 \quad (\text{PCROP area end address 0x0800 13FF})\]

Data read access to a PCROP-protected address raises the RDERR flag.

PCROP-protected addresses are also write protected. Write access to a PCROP-protected address raises the WRPERR flag.

PCROP-protected areas are also erase protected. Attempts to erase a page including at least one PCROP-protected subpage fails. Moreover, software mass erase cannot be performed if a PCROP-protected area is defined.

Deactivation of PCROP can only occur upon the RDP change from Level 1 to Level 0. Modification of user options to clear PCROP or to decrease the size of a PCROP-protected area do not have any effect to the PCROP areas. On the contrary, it is possible to increase the size of the PCROP-protected areas.

With the option bit PCROP_RDP cleared, the change of RDP from Level 1 to Level 0 triggers a partial mass erase that preserves the contents of the Flash memory pages overlapping with PCROP-protected areas. Refer to section Changing the read protection level for details.
Note: With PCROP_RDP cleared, it is recommended to either define the PCROP area start and end onto Flash memory page boundaries (2-Kbyte granularity), or to keep reserved and empty the PCROP-unprotected memory space of the PCROP area boundary pages (pages inside which the PCROP area starts and ends).

### 3.5.3 FLASH write protection (WRP)

The user area in Flash memory can be protected against unwanted write operations. Two write-protected (WRP) areas can be defined, with page (2-Kbyte) granularity. Each area is defined by a start page offset and an end page offset related to the physical Flash memory base address. These offsets are defined in the WRP address registers FLASH WRP area A address register (FLASH_WRP1AR) and FLASH WRP area B address register (FLASH_WRP1BR).

The WRP "y" area (y=A, B) is defined from the address

\[
\text{Flash memory Base address} + \left[\text{WRP1y\_STRT x 0x0800}\right] \text{ (included)}
\]

to the address

\[
\text{Flash memory Base address} + \left[\left(\text{WRP1y\_END+1}\right) x 0x0800\right] \text{ (excluded)}.
\]

The minimum WRP area size is one WRP page (2 Kbytes):

\[
\text{WRP1y\_END} = \text{WRP1y\_STRT}.
\]

For example, to protect by WRP from the address 0x0800 1000 (included) to the address 0x0800 3FFF (included):

- If boot in Flash memory is selected, FLASH_WRP1AR register must be programmed with:
  - WRP1A\_STRT = 0x02.
  - WRP1A\_END = 0x07.
- WRP1B\_STRT and WRP1B\_END in FLASH_WRP1BR can be used instead (area B in Flash memory).

When WRP is active, it cannot be erased or programmed. Consequently, a software mass erase cannot be performed if one area is write-protected.

If an erase/program operation to a write-protected part of the Flash memory is attempted, the write protection error flag (WRPERR) of the FLASH_SR register is set. This flag is also set for any write access to:

- OTP area
- part of the Flash memory that can never be written like the ICP
- PCROP area

<table>
<thead>
<tr>
<th>PCROP register values (x=A or B)</th>
<th>PCROP-protected area</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCROP1x_STRT = PCROP1x_END</td>
<td>Full Flash memory</td>
</tr>
<tr>
<td>PCROP1x_STRT &gt; PCROP1x_END</td>
<td>None (unprotected)</td>
</tr>
<tr>
<td>PCROP1x_STRT &lt; PCROP1x_END</td>
<td>Subpages from PCROP1x_STRT to PCROP1x_END (read-, write-, and erase-protected); PCROP area boundary pages (erase-protected).</td>
</tr>
</tbody>
</table>

Table 15: PCROP protection
Note: When the Flash memory read protection level is selected (RDP level = 1), it is not possible to program or erase the memory if the CPU debug features are connected (single wire) or boot code is being executed from SRAM or system Flash memory, even if WRP is not activated. Any attempt generates a hard fault (BusFault).

Table 16: WRP protection

<table>
<thead>
<tr>
<th>WRP registers values (x=A or B)</th>
<th>WRP-protected area</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRP1x_STRT = WRP1x_END</td>
<td>Page WRP1x</td>
</tr>
<tr>
<td>WRP1x_STRT &gt; WRP1x_END</td>
<td>None (unprotected)</td>
</tr>
<tr>
<td>WRP1x_STRT &lt; WRP1x_END</td>
<td>Pages from WRP1x_STRT to WRP1x_END</td>
</tr>
</tbody>
</table>

Note: To validate the WRP options, the option bytes must be reloaded by setting the OBL_LAUNCH bit in Flash memory control register.

3.5.4 Securable memory area

The main purpose of the securable memory area is to protect a specific part of Flash memory against undesired access. After system reset, the code in the securable memory area can only be executed until the securable area becomes secured and never again until the next system reset. This allows implementing software security services such as secure key storage or safe boot.

Securable memory area is located in the Main Flash memory. It is dedicated to executing trusted code. When not secured, the securable memory behaves like the rest of Main Flash memory. When secured (the SEC_PROT bit of the FLASH_CR register set), any access to securable memory area (fetch, read, programming, erase) is rejected, generating a bus error. The securable area can only be unsecured by a system reset.

The size of the securable memory area is defined by the SEC_SIZE[6:0] bitfield of the FLASH_SECUR register. It can be modified only in RDP Level 0. Its content is erased upon changing from RDP Level 1 to Level 0. Its content is erased even if it overlaps with PCROP subpages.

Note: Before activating the securable memory area, move the vector table outside this area if necessary.

Note: Upon change from RDP Level 1 to Level 0 while the PCROP_RDP bit is cleared, the securable memory area is erased even if it overlaps with the PCROP subpages. The PCROP subpages not overlapping with the securable memory area are not erased. See Table 17.

Table 17. Securable memory erase at RDP Level 1 to Level 0 change

<table>
<thead>
<tr>
<th>Securable memory size (SEC_SIZE[6:0])</th>
<th>PCROP_RDP</th>
<th>Erased pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>All (mass erase)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>All but PCROP</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>1</td>
<td>All (mass erase)</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>0</td>
<td>All but PCROP outside the securable memory area</td>
</tr>
</tbody>
</table>
3.5.5 Disabling core debug access

For executing sensitive code or manipulating sensitive data in securable memory area, the debug access to the core can temporarily be disabled. 

*Figure 4* gives an example of managing DBG_SWEN and SEC_PROT bits.

![Figure 4. Example of disabling core debug access](MSv42192V1)

3.5.6 Forcing boot from Flash memory

To increase the security and establish a chain of trust, the BOOT_LOCK option bit of the FLASH_SECR register allows forcing the system to boot from the Main Flash memory regardless the other boot options. It is always possible to set the BOOT_LOCK bit. However, it is possible to reset it only when:

- RDP is set to Level 0, or
- RDP is set to Level 1, while Level 0 is requested and a full mass-erase is performed.

3.6 FLASH interrupts

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Event flag/interrupt clearing method</th>
<th>Interrupt enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>End of operation</td>
<td>EOP&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>Write EOP=1</td>
<td>EOPIE</td>
</tr>
<tr>
<td>Operation error</td>
<td>OPERR&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>Write OPERR=1</td>
<td>ERRIE</td>
</tr>
<tr>
<td>Read protection error</td>
<td>RDERR</td>
<td>Write RDERR=1</td>
<td>RDERRIE</td>
</tr>
<tr>
<td>Write protection error</td>
<td>WRPERR</td>
<td>Write WRPERR=1</td>
<td>N/A</td>
</tr>
<tr>
<td>Size error</td>
<td>SIZERR</td>
<td>Write SIZERR=1</td>
<td>N/A</td>
</tr>
<tr>
<td>Programming sequential error</td>
<td>PROGERR</td>
<td>Write PROGERR=1</td>
<td>N/A</td>
</tr>
<tr>
<td>Programming alignment error</td>
<td>PGAERR</td>
<td>Write PGAERR=1</td>
<td>N/A</td>
</tr>
<tr>
<td>Programming sequence error</td>
<td>PGSERR</td>
<td>Write PGSERR=1</td>
<td>N/A</td>
</tr>
<tr>
<td>Data miss during fast programming error</td>
<td>MISSERR</td>
<td>Write MISSERR=1</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Table 18. FLASH interrupt requests (continued)

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Event flag/interrupt clearing method</th>
<th>Interrupt enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast programming error</td>
<td>FASTERR</td>
<td>Write FASTERR=1</td>
<td>N/A</td>
</tr>
<tr>
<td>ECC error correction</td>
<td>ECCC</td>
<td>Write ECCC=1</td>
<td>ECCCIE</td>
</tr>
<tr>
<td>ECC double error (NMI)</td>
<td>ECCD</td>
<td>Write ECCD=1</td>
<td>N/A</td>
</tr>
</tbody>
</table>

1. EOP is set only if EOPIE is set.
2. OPERR is set only if ERRIE is set.
3.7  FLASH registers

3.7.1  FLASH access control register (FLASH_ACR)

Address offset: 0x000
Reset value: 0x0004 0600

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

Bits 31:19  Reserved, must be kept at reset value

Bit 18  **DBG_SWEN**: Debug access software enable
Software may use this bit to enable/disable the debugger read access.
0: Debugger disabled
1: Debugger enabled

Bit 17  Reserved, must be kept at reset value

Bit 16  **EMPTY**: Main Flash memory area empty
This bit indicates whether the first location of the Main Flash memory area is erased or has a programmed value.
0: Main Flash memory area empty
1: Main Flash memory area programmed
The bit can be set and reset by software..

Bits 15:12  Reserved, must be kept at reset value

Bit 11  **ICRST**: CPU Instruction cache reset
0: CPU Instruction cache is not reset
1: CPU Instruction cache is reset
This bit can be written only when the instruction cache is disabled.

Bit 10  Reserved, must be kept at reset value

Bit 9  **ICEN**: CPU Instruction cache enable
0: CPU Instruction cache is disabled
1: CPU Instruction cache is enabled
Bit 8  **PRFTEN**: CPU Prefetch enable  
0: CPU Prefetch disabled  
1: CPU Prefetch enabled  

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

Bits 2:0  **LATENCY[2:0]**: Flash memory access latency  
The value in this bitfield represents the ratio of the HCLK clock period to the Flash memory access time.  
000: Zero wait states  
001: One wait state  
010: Two wait states  
Others: Reserved  
A a new write into the bitfield becomes effective when it returns the same value upon read.

### 3.7.2  FLASH key register (FLASH_KEYR)  
Address offset: 0x008  
Reset value: 0x0000 0000

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

**KEY[31:16]**

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

**KEY[15:0]**

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

Bits 31:0  **KEY[31:0]**: FLASH key  
The following values must be written consecutively to unlock the *FLASH control register (FLASH_CR)*, thus enabling programming/erasing operations:  
KEY1: 0x4567 0123  
KEY2: 0xCDEF 89AB

### 3.7.3  FLASH option key register (FLASH_OPTKEYR)  
Address offset: 0x00C  
Reset value: 0x0000 0000

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

**OPTKEY[31:16]**

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

**OPTKEY[15:0]**

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

Bits 31:0  **OPTKEY[31:0]**: Option byte key  
The following values must be written consecutively to unlock the Flash memory option registers, enabling option byte programming/erasing operations:  
KEY1: 0x0819 2A3B  
KEY2: 0x4C5D 6E7F
3.7.4 FLASH status register (FLASH_SR)

Address offset: 0x010
Reset value: 0x0000 X 0000

<table>
<thead>
<tr>
<th>Bit 31:19</th>
<th>Reserved, must be kept at reset value</th>
</tr>
</thead>
</table>

Bit 18 `CFGBSY`: Programming or erase configuration busy.
This flag is set and reset by hardware. (set when first word is sent and reset when program operation completes or is interrupted by an error.)
When set to 1 the programming and erase settings in PB and PNB bits requested by `FLASH control register (FLASH_CR)` are used (busy), and cannot be changed (a programming or erase operation is ongoing).
When reset to 0 programming and erase settings in PB and PNB bits in `FLASH control register (FLASH_CR)` can be modified.

Bit 17 Reserved, must be kept at reset value

Bit 16 `BSY1`: Busy
This flag indicates that a Flash memory operation requested by `FLASH control register (FLASH_CR)` is in progress. This bit is set at the beginning of a Flash memory operation, and reset when the operation finishes or when an error occurs.

Bit 15 `OPTVERR`: Option and Engineering bits loading validity error
Set by hardware when the options and engineering bits read may not be the one configured by the user or production. If options and engineering bits haven’t been properly loaded, OPTVERR is set again after each system reset. Option bytes that fail loading will be forced to a safe value, see Section 3.4.2: FLASH option byte programming.
Cleared by writing 1.

Bit 14 `RDERR`: PCROP read error
Set by hardware when an address to be read belongs to a read protected area of the Flash memory (PCROP protection). An interrupt is generated if RDERRIE is set in FLASH_CR.
Cleared by writing 1.

Bits 13:10 Reserved, must be kept at reset value

Bit 9 `FASTERR`: Fast programming error
Set by hardware when a fast programming sequence (activated by FSTPG) is interrupted due to an error (alignment, size, write protection or data miss). The corresponding status bit (PGAERR, SIZERR, WRPERR or MISSERR) is set at the same time.
Cleared by writing 1.

Bit 8 `MISERR`: Fast programming data miss error
In Fast programming mode, 32 double words (256 bytes) must be sent to Flash memory successively, and the new data must be sent to the logic control before the current data is fully programmed. MISSERR is set by hardware when the new data is not present in time.
Cleared by writing 1.
3.7.5 **FLASH control register (FLASH_CR)**

Address offset: 0x014

Reset value: 0xC000 0000

Access: no wait state when no Flash memory operation is on going, word, half-word and byte access

This register cannot be modified when CFGBSY in **FLASH status register (FLASH_SR)** is set.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>LOCK</td>
<td>rs</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>OPT</td>
<td>rs</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>LOCK</td>
<td>rs</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Res.</td>
<td>rs</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>SEC. PROT.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>OBL. LAUNCH</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>RD</td>
<td>nw</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>ERRIE</td>
<td>nw</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>EOPIE</td>
<td>nw</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Res.</td>
<td>nw</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Res.</td>
<td>nw</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Res.</td>
<td>nw</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Res.</td>
<td>nw</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Res.</td>
<td>nw</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>FSTPG</td>
<td>nw</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>OPT STRT</td>
<td>rs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STRT</td>
<td>rs</td>
<td></td>
</tr>
</tbody>
</table>

**Bit 7**: `PGSERR`: Programming sequence error
Set by hardware when a write access to the Flash memory is performed by the code while PG or FSTPG have not been set previously. Set also by hardware when PROGERR, SIZERR, PGAERR, WRPERR, MISSERR or FASTERR is set due to a previous programming error.
Cleared by writing 1.

**Bit 6**: `SIZERR`: Size error
Set by hardware when the size of the access is a byte or half-word during a program or a fast program sequence. Only double word programming is allowed (consequently: word access).
Cleared by writing 1.

**Bit 5**: `PGAERR`: Programming alignment error
Set by hardware when the data to program cannot be contained in the same double word (64-bit) Flash memory in case of standard programming, or if there is a change of page during fast programming.
Cleared by writing 1.

**Bit 4**: `WRPERR`: Write protection error
Set by hardware when an address to be erased/programmed belongs to a write-protected part (by WRP, PCROP or RDP Level 1) of the Flash memory.
Cleared by writing 1.

**Bit 3**: `PROGERR`: Programming error
Set by hardware when a double-word address to be programmed contains a value different from '0xFFFF FFFF' before programming, except if the data to write is '0x0000 0000'.
Cleared by writing 1.

**Bit 2**: Reserved, must be kept at reset value

**Bit 1**: `OPERR`: Operation error
Set by hardware when a Flash memory operation (program / erase) completes unsuccessfully.
This bit is set only if error interrupts are enabled (ERRIE=1).
Cleared by writing ‘1’.

**Bit 0**: `EOP`: End of operation
Set by hardware when one or more Flash memory operation (programming / erase) has been completed successfully.
This bit is set only if the end of operation interrupts are enabled (EOPIE=1).
Cleared by writing 1.
### Embedded Flash memory (FLASH) RM0444

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>LOCK: FLASH_CR Lock</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set only. When set, the FLASH_CR register is locked. It is cleared by hardware after detecting the unlock sequence. In case of an unsuccessful unlock operation, this bit remains set until the next system reset.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>OPTLOCK: Options Lock</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set only. When set, all bits concerning user option in FLASH_CR register and so option page are locked. This bit is cleared by hardware after detecting the unlock sequence. The LOCK bit must be cleared before doing the unlock sequence for OPTLOCK bit. In case of an unsuccessful unlock operation, this bit remains set until the next reset.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>Reserved, must be kept at reset value</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>SEC_PROT: Securable memory area protection enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit enables the protection on securable area, provided that a not-null securable memory area size (SEC_SIZE) has been defined in option bytes.</td>
</tr>
<tr>
<td></td>
<td>0: Content of securable area accessible</td>
</tr>
<tr>
<td></td>
<td>1: Content of securable area not accessible</td>
</tr>
<tr>
<td></td>
<td>Software may only set it to 1. It may only be reset by a system reset.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 27</th>
<th>OBL_LAUNCH: Forces the option byte loading</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When set to 1, this bit forces the option byte reloading. This bit is cleared only when the option byte loading is complete. It cannot be written if OPTLOCK is set.</td>
</tr>
<tr>
<td></td>
<td>0: Option byte loading complete</td>
</tr>
<tr>
<td></td>
<td>1: Option byte loading requested</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 26</th>
<th>RDERRIE: PCROP read error interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit enables the interrupt generation when the RDERR bit of the FLASH_SR register is set.</td>
</tr>
<tr>
<td></td>
<td>0: PCROP read error interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: PCROP read error interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 25</th>
<th>ERRIE: Error interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit enables the interrupt generation when the OPERR bit of the FLASH_SR register is set.</td>
</tr>
<tr>
<td></td>
<td>0: OPERR error interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: OPERR error interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 24</th>
<th>EOPIE: End of operation interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit enables the interrupt generation when the EOP bit of the FLASH_SR register is set.</td>
</tr>
<tr>
<td></td>
<td>0: EOP Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: EOP Interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 23:19</th>
<th>Reserved, must be kept at reset value</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 18</th>
<th>FSTPG: Fast programming</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: Fast programming disabled</td>
</tr>
<tr>
<td></td>
<td>1: Fast programming enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 17</th>
<th>OPTSTRT: Start of modification of option bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit triggers an options operation when set.</td>
</tr>
<tr>
<td></td>
<td>This bit is set only by software, and is cleared when the BSY1 bit is cleared in FLASH_SR.</td>
</tr>
</tbody>
</table>
Bit 16 **STRT**: Start
This bit triggers an erase operation when set. If MER1 and PER bits are reset and the STRT bit is set, an unpredictable behavior may occur without generating any error flag. This condition should be forbidden.
This bit is set only by software, and is cleared when the BSY1 bit is cleared in FLASH_SR.

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

Bits 8:3 **PNB[5:0]**: Page number selection
These bits select the page to erase:
0x00: page 0
0x01: page 1
...
0x3F: page 63

Bit 2 **MER1**: Mass erase
This bit triggers the mass erase (all user pages) when set.

Bit 1 **PER**: Page erase
0: page erase disabled
1: page erase enabled

Bit 0 **PG**: Programming
0: Flash memory programming disabled
1: Flash memory programming enabled

### 3.7.6 FLASH ECC register (FLASH_ECCR)

Address offset: 0x018
Reset value: 0x0000 0000

Access: no wait state when no Flash memory operation is on going, word, half-word and byte access

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>nw</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res.</td>
<td>Res.</td>
<td>ADDR_ECC[13:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **ECCD**: ECC detection
Set by hardware when two ECC errors have been detected. When this bit is set, a NMI is generated.
Cleared by writing 1.

Bit 30 **ECC**: ECC correction
Set by hardware when one ECC error has been detected and corrected. An interrupt is generated if ECCIE is set.
Cleared by writing 1.

Bits 29:25 Reserved, must be kept at reset value
3.7.7 **FLASH option register (FLASH_OPTR)**

Address offset: 0x020

Reset value: 0b11XX XXXX 1X11 XXXXX XXXXX XXXXX XXXXX XXXXX. The option bits are loaded with values from Flash memory at power-on reset release.

Access: no wait state when no Flash memory operation is on going, word, half-word and byte access

| Bit 31:30 | Reserved, must be kept at reset value |
| Bit 29 | IRHEN: Internal reset holder enable bit |
| 0 | Internal resets are propagated as simple pulse on NRST pin |
| 1 | Internal resets drives NRST pin low until it is seen as low level |
| Bit 28:27 | NRST_MODE[1:0] |
| 00 | Reserved |
| 01 | Reset input only: a low level on the NRST pin generates system reset, internal RESET not propagated to the NSRT pin |
| 10 | GPIO: standard GPIO pad functionality, only internal RESET possible |
| 11 | Bidirectional reset: NRST pin configured in reset input/output mode (legacy mode) |
| Bit 26 | nBOOT0: nBOOT0 option bit |
| 0 | nBOOT0=0 |
| 1 | nBOOT0=1 |
| Bit 25 | nBOOT1: Boot configuration |
| Together with the BOOT0 pin or option bit nBOOT0 (depending on nBOOT_SEL option bit configuration), this bit selects boot mode from the Main Flash memory, SRAM or the System memory. Refer to **Section 2.5: Boot configuration** |
Bit 24  nBOOT_SEL
  0: BOOT0 signal is defined by BOOT0 pin value (legacy mode)
  1: BOOT0 signal is defined by nBOOT0 option bit
Bit 23  Reserved, must be kept at reset value
Bit 22  RAM_PARITY_CHECK: SRAM parity check control
  0: SRAM parity check enable
  1: SRAM parity check disable
Bits 21:20  Reserved, must be kept at reset value
Bit 19  WWDG_SW: Window watchdog selection
  0: Hardware window watchdog
  1: Software window watchdog
Bit 18  IWDG_STDBY: Independent watchdog counter freeze in Standby mode
  0: Independent watchdog counter is frozen in Standby mode
  1: Independent watchdog counter is running in Standby mode
Bit 17  IWDG_STOP: Independent watchdog counter freeze in Stop mode
  0: Independent watchdog counter is frozen in Stop mode
  1: Independent watchdog counter is running in Stop mode
Bit 16  IDWG_SW: Independent watchdog selection
  0: Hardware independent watchdog
  1: Software independent watchdog
Bit 15  nRSTS_SHDW
  0: Reset generated when entering the Shutdown mode
  1: No reset generated when entering the Shutdown mode
Bit 14  nRST_STDBY
  0: Reset generated when entering the Standby mode
  1: No reset generated when entering the Standby mode
Bit 13  nRST_STOP
  0: Reset generated when entering the Stop mode
  1: No reset generated when entering the Stop mode
Bits 12:11  BORR_LEV[1:0]: These bits contain the $V_{DD}$ supply level threshold that releases the reset.
  00: BOR rising level 1 with threshold around 2.1 V
  01: BOR rising level 2 with threshold around 2.3 V
  01: BOR rising level 3 with threshold around 2.6 V
  11: BOR rising level 4 with threshold around 2.9 V
Bits 10:9  BORF_LEV[1:0]: These bits contain the $V_{DD}$ supply level threshold that activates the reset.
  00: BOR falling level 1 with threshold around 2.0 V
  01: BOR falling level 2 with threshold around 2.2 V
  01: BOR falling level 3 with threshold around 2.5 V
  11: BOR falling level 4 with threshold around 2.8 V
3.7.8 FLASH PCROP area A start address register (FLASH_PCROP1ASR)

Address offset: 0x024

Reset value: ‘b1111 1111 1111 1111 1111 1111 XXXX XXXX. The option bits are loaded with values from Flash memory at power-on reset release.

Access: no wait state when no Flash memory operation is on going, word, half-word access

<table>
<thead>
<tr>
<th>Bit 31-8</th>
<th>Bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
3.7.10 **FLASH WRP area A address register (FLASH_WRP1AR)**

Address offset: 0x02C

Reset value: 0xFFXX FFXX. The option bits are loaded with values from Flash memory at power-on reset release.

Access: no wait state when no Flash memory operation is on going, word, half-word and byte access.

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRP1A_END[5:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>WRP1A_STRT[5:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **WRP1A_END[5:0]**: WRP area A end offset

This bitfield contains the offset of the last page of the WRP area A.(1)

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

Bits 5:0 **WRP1A_STRT[5:0]**: WRP area A start offset

This bitfield contains the offset of the first page of the WRP area A.(1)

1. The number of effective bits depends on the size of Flash memory in the device.
3.7.11  FLASH WRP area B address register (FLASH_WRP1BR)

Address offset: 0x030

Reset value: 0xFFXX FFXX. The option bits are loaded with values from Flash memory at power-on reset release.

Access: no wait state when no Flash memory operation is on going, word, half-word and byte access.

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

**WRP1B_END[5:0]**

Bits 31:22: Reserved, must be kept at reset value.

Bits 21:16 **WRP1B_END[5:0]**: WRP area B end offset
This bitfield contains the offset of the last page of the WRP area B.\(^{(1)}\)

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

Bits 5:0 **WRP1B_STRT[5:0]**: WRP area B start offset
This bitfield contains the offset of the first page of the WRP area B.\(^{(1)}\)

---

3.7.12  FLASH PCROP area B start address register (FLASH_PCROP1BSR)

Address offset: 0x034

Reset value: 'b1111 1111 1111 1111 1111 111X XXXX XXXX. The option bits are loaded with values from Flash memory at power-on reset release.

Access: no wait state when no Flash memory operation is on going, word, half-word access

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

**PCROP1B_STRT[7:0]**

Bits 31:8: Reserved, must be kept cleared

Bits 7:0 **PCROP1B_STRT[7:0]**: PCROP1B area start offset
This bitfield contains the offset of the first subpage of the PCROP1B area.
3.7.13 FLASH PCROP area B end address register (FLASH_PCROP1BER)

Address offset: 0x038

Reset value: 'b1111 1111 1111 1111 1111 111X XXXX XXXX. The option bits are loaded with values from Flash memory at power-on reset release.

Access: no wait state when no Flash memory operation is on going, word, half-word access

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

Bits 31:8 Reserved, must be kept cleared

Bits 7:0 PCROP1B_END[7:0]: PCROP1B area end offset
Contains the offset of the last subpage of the PCROP1B area.

3.7.14 FLASH security register (FLASH_SEC)

Address offset: 0x080

Reset value: 0x0000 0000. The option bits are loaded with values from Flash memory at power-on reset release.

Access: no wait state when no Flash memory operation is on going, word, half-word access

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

Bits 31:17 Reserved, must be kept cleared

Bit 16 BOOT_LOCK: used to force boot from user area
0: Boot based on the pad/option bit configuration
1: Boot forced from Main Flash memory

Bits 15:7 Reserved, must be kept cleared

Bits 6:0 SEC_SIZE[6:0]: Securable memory area size
Contains the number of securable Flash memory pages
## 3.7.15 FLASH register map

Table 19. FLASH register map and reset values

| Offset | Register     | Field  | Zn | Sn | Kn | Ln | Rn | E | EOE | Recent | Latency [2:0] | Reset value | Offset | Register | Field  | Zn | Sn | Kn | Ln | Rn | E | EOE | Recent | Latency [2:0] | Reset value | Offset | Register | Field  |
|--------|--------------|--------|----|----|----|----|----|---|----|--------|--------------|-------------|---------|---------|---------|--------|----|----|----|----|----|---|----|--------|--------------|-------------|---------|---------|---------|
| 0x00   | FLASH_ACR    |        |    |    |    |    |    | X |    |        |              |             |         |         |         |        |    |    |    |    |    |   |    |        |              |             |         |         |         |        |
| 0x04   | FLASH_KEYR   | KEYR[31:0] |    |    |    |    |    | X |    |        |              |             |         |         |         |        |    |    |    |    |    |   |    |        |              |             |         |         |         |        |
| 0x08   | FLASH_OPTKEYR| OPTKEYR[31:0] |    |    |    |    |    | X |    |        |              |             |         |         |         |        |    |    |    |    |    |   |    |        |              |             |         |         |         |        |
| 0x10   | FLASH_SR     |        |    |    |    |    |    | X |    |        |              |             |         |         |         |        |    |    |    |    |    |   |    |        |              |             |         |         |         |        |
| 0x14   | FLASH_CR     |        |    |    |    |    |    | X |    |        |              |             |         |         |         |        |    |    |    |    |    |   |    |        |              |             |         |         |         |        |
| 0x18   | FLASH_ECCR   |        |    |    |    |    |    | X |    |        |              |             |         |         |         |        |    |    |    |    |    |   |    |        |              |             |         |         |         |        |
| 0x20   | FLASH_OPTR   |        |    |    |    |    |    | X |    |        |              |             |         |         |         |        |    |    |    |    |    |   |    |        |              |             |         |         |         |        |
| 0x24   | FLASH_PCROP1ASR | PCROP1A_STRT[7:0] |    |    |    |    |    | X |    |        |              |             |         |         |         |        |    |    |    |    |    |   |    |        |              |             |         |         |         |        |
| 0x28   | FLASH_PCROP1AER | PCROP1A_END[7:0] |    |    |    |    |    | X |    |        |              |             |         |         |         |        |    |    |    |    |    |   |    |        |              |             |         |         |         |        |
| 0x2C   | FLASH_WRP1AAR | WRP1A_END[5:0] |    |    |    |    |    | X |    |        |              |             |         |         |         |        |    |    |    |    |    |   |    |        |              |             |         |         |         |        |
| 0x30   | FLASH_WRP1BR | WRP1B_END[5:0] |    |    |    |    |    | X |    |        |              |             |         |         |         |        |    |    |    |    |    |   |    |        |              |             |         |         |         |        |
| 0x34   | FLASH_PCP1BBR | WRP1B_STRT[7:0] |    |    |    |    |    | X |    |        |              |             |         |         |         |        |    |    |    |    |    |   |    |        |              |             |         |         |         |        |

---

**Notes:**
- Zn, Sn, Kn, Ln, Rn, E, EOE, Recent, Latency: Register fields.
- Reset value: The value that each register is initialized to upon reset.
- Offset: The memory offset for each register.

**Legend:**
- XX...X: Indicates the value of the field.
- X: Indicates a specific value for the field.
- [7:0]: Indicates the field is 8 bits wide.
Table 19. FLASH register map and reset values (continued)

| Offset   | Register   | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x038    | FLASH_PCROP1BERR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x03C - 0x07F | Reserved   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x080    | FLASH_SECRR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.2 on page 54 for the register boundary addresses.
4 Power control (PWR)

4.1 Power supplies

The STM32G0x1 devices require a 1.7 V to 3.6 V operating supply voltage (VDD). Several different power supplies are provided to specific peripherals:

- **V\text{DD} = 1.7 V (1.60 V) to 3.6 V**
  
  V\text{DD} is the external power supply for the internal regulator and the system analog such as reset, power management and internal clocks. It is provided externally through VDD/VDDA pin.

  Note that the minimum voltage of 1.7 V corresponds to power-on reset release threshold VPOR\text{(MAX)}. Once this threshold is crossed and power-on reset is released, the functionality is guaranteed down to power-down reset threshold VPDR\text{(MIN)}.

- **V\text{DDA} = 1.62 V (ADC and COMP) / 1.8 V (DAC) / 2.4 V (VREFBUF) to 3.6 V**
  
  V\text{DDA} is the analog power supply for the A/D converter, D/A converters, voltage reference buffer and comparators. V\text{DDA} voltage level is identical to V\text{DD} voltage as it is provided externally through VDD/VDDA pin.

- **V\text{DDIO1} = V\text{DD}**

  V\text{DDIO1} is the power supply for the I/Os. V\text{DDIO1} voltage level is identical to V\text{DD} voltage as it is provided externally through VDD/VDDA pin.

- **V\text{BAT} = 1.55 V to 3.6 V**

  V\text{BAT} is the power supply (through a power switch) for RTC, TAMP, low-speed external 32.768 kHz oscillator and backup registers when V\text{DD} is not present. V\text{BAT} is provided externally through VBAT pin. When this pin is not available on the package, it is internally bonded to VDD/VDDA.

- **VREF\text{+}** is the input reference voltage for the ADC and DAC, or the output of the internal voltage reference buffer (when enabled). When V\text{DDA} < 2 V, VREF\text{+} must be equal to V\text{DDA}. When V\text{DDA} ≥ 2 V, VREF\text{+} must be between 2 V and V\text{DDA}. It can be grounded when the ADC and DAC are not active.

  The internal voltage reference buffer supports two output voltages, which is configured with VRS bit of the VREFBUF\_CSR register:

  - VREF\text{+} around 2.048 V (requiring V\text{DDA} equal to or higher than 2.4 V)
  - VREF\text{+} around 2.5 V (requiring V\text{DDA} equal to or higher than 2.8 V)

  VREF\text{+} is delivered through VREF\text{+} pin. On packages without VREF\text{+} pin, VREF\text{+} is internally connected with V\text{DD}, and the internal voltage reference buffer must be kept disabled (refer to datasheets for package pinout description).

- **V\text{CORE}**

  An embedded linear voltage regulator is used to supply the V\text{CORE} internal digital power. V\text{CORE} is the power supply for digital peripherals, SRAM and Flash memory. The Flash memory is also supplied by V\text{DD}.
4.1.1 **ADC and DAC reference voltage**

To ensure a better accuracy on low-voltage inputs and outputs, the user can connect to $V_{REF+}$ a separate reference voltage lower than $V_{DDA}$. $V_{REF+}$ is the highest voltage, represented by the full scale value, for an analog input (ADC) or output (DAC) signal.

$V_{REF+}$ can be provided either by an external reference of by an internal buffered voltage reference (VREFBUF).

The internal buffered voltage reference is enabled by setting the ENVR bit in the VREFBUF control and status register (VREFBUF_CSR). The internal buffered voltage reference is set to 2.5 V when the VRS bit is set and to 2.048 V when the VRS bit is cleared. The internal buffered voltage reference can also provide the voltage to external components through $V_{REF+}$ pin. Refer to the device datasheet and to Section 16: Voltage reference buffer (VREFBUF) for further information.

4.1.2 **Battery backup of RTC domain**

To retain the content of the backup registers and supply the RTC and TAMP functions when $V_{DD}$ is turned off, the VBAT pin can be connected to an optional backup voltage supplied by a battery or by another source.

The VBAT pin powers the RTC and TAMP units, the LSE oscillator and the PC13 to PC15 I/Os, allowing the RTC and TAMP to operate even when the main power supply is turned off. The switch to the $V_{BAT}$ supply is controlled by the power-down reset embedded in the Reset block.
Warning: During \( t_{\text{RSTTEMPO}} \) (temporization at \( V_{\text{DD}} \) startup) or after a PDR has been detected, the power switch between \( V_{\text{BAT}} \) and \( V_{\text{DD}} \) remains connected to \( V_{\text{BAT}} \). During the startup phase, if \( V_{\text{DD}} \) is established in less than \( t_{\text{RSTTEMPO}} \) (refer to the datasheet for the value of \( t_{\text{RSTTEMPO}} \)) and \( V_{\text{DD}} > V_{\text{BAT}} + 0.6 \) V, a current may be injected into \( V_{\text{BAT}} \) through an internal diode connected between \( V_{\text{DD}} \) and the power switch (\( V_{\text{BAT}} \)). If the power supply/battery connected to the \( V_{\text{BAT}} \) pin cannot support this current injection, it is recommended to connect an external low-drop diode between this power supply and the \( V_{\text{BAT}} \) pin.

If no external battery is used in the user application, it is recommended to connect \( V_{\text{BAT}} \) pin externally to \( V_{\text{DD}}/V_{\text{DDA}} \) pin with a 100 nF external ceramic decoupling capacitor.

When the RTC domain is supplied by \( V_{\text{DD}} \) (power switch connected to \( V_{\text{DD}} \)), all the related pin functions are available:

When the RTC domain is supplied by \( V_{\text{BAT}} \) (power switch connected to \( V_{\text{BAT}} \) because \( V_{\text{DD}} \) is not present), only the following functions are available:

- PC13, PC14 and PC15 can be controlled only by RTC, TAMP or LSE (refer to Section 29.3: RTC functional description)
- RTC_OUT1 function on PC13
- RTC_TS function on PC13 or PA4
- TAMP_IN1 function on PC13 or PA4 and TAMP_IN2 function on PA0

Note: Due to the fact that the power switch can transfer only a limited amount of current (3 mA), the use of GPIO PC13 to PC15 in output mode is restricted: the speed has to be limited to 2 MHz with a maximum load of 30 pF and these I/Os must not be used as a current source (e.g. to drive a LED).

RTC domain access

After a system reset, the RTC domain (RTC registers and backup registers) is protected against possible unwanted write accesses. To enable access to the RTC domain, proceed as follows:

1. Enable the power interface clock by setting the PWREN bits of the APB peripheral clock enable register 1 (RCC_APBENR1).
2. Set the DBP bit of the Power control register 1 (PWR_CR1) to enable access to the RTC domain.
3. Select the RTC clock source in the RTC domain control register (RCC_BDCR).
4. Enable the RTC clock by setting the RTCEN bit in the RTC domain control register (RCC_BDCR).
VBAT battery charging

When \( V_{DD} \) is present, it is possible to charge the external battery on VBAT through an internal resistance.

The VBAT charging is done either through a 5 k\( \Omega \) resistor or through a 1.5 k\( \Omega \) resistor depending on the VBRS bit value in the PWR_CR4 register.

The battery charging is enabled by setting VBE bit in the PWR_CR4 register. It is automatically disabled in VBAT mode.

4.1.3 Voltage regulator

Two embedded linear voltage regulators supply all the digital circuitries, except for the Standby circuitry and the RTC domain. The main regulator output voltage (\( V_{CORE} \)) can be programmed by software to two different power ranges (Range 1 and Range 2) in order to optimize the consumption depending on the system maximum operating frequency (refer to Section 5.2.7: Clock source frequency versus voltage scaling and to Section 3.3.4: FLASH read access latency).

The voltage regulators are always enabled after a reset. Depending on the user application modes, the \( V_{CORE} \) supply is provided either by the main regulator (MR) or by the low-power regulator (LPR).

- In Run, Sleep and Stop 0 modes, both regulators are enabled and the main regulator (MR) supplies full power to the \( V_{CORE} \) domain (core, memories and digital peripherals).
- In Low-power run and Low-power sleep modes, the main regulator is off and the low-power regulator (LPR) supplies low-power to the \( V_{CORE} \) domain, preserving the contents of the registers and SRAM.
- In Stop 1 mode, the main regulator is off and the low-power regulator (LPR) supplies low-power to the \( V_{CORE} \) domain, preserving the contents of the registers and SRAM.
- In Standby mode with SRAM content preserved (RRS bit is set in the PWR_CR3 register), the main regulator (MR) is off and the low-power regulator (LPR) provides the supply to SRAM only. The core and digital peripherals (except Standby circuitry and RTC domain) are powered off.
- In Standby mode, both regulators are powered off. The contents of the registers and SRAM is lost except for the Standby circuitry and the RTC domain.
- In Shutdown mode, both regulators are powered off. When exiting Shutdown mode, a power-on reset is generated. Consequently, the contents of the registers and SRAM is lost, except for the RTC domain.
4.1.4 Dynamic voltage scaling management

The dynamic voltage scaling is a power management technique which consists in increasing or decreasing the voltage used for the digital peripherals (V\text{CORE}), according to the application performance and power consumption needs.

Dynamic voltage scaling to increase V\text{CORE} is known as overvolting. It allows to improve the device performance.

Dynamic voltage scaling to decrease V\text{CORE} is known as undervolting. It is performed to save power, particularly in laptop and other mobile devices where the energy comes from a battery and is thus limited.

Two voltage ranges are available:

- **Range 1: High-performance range**
  
The main regulator provides a typical output voltage at 1.2 V. The system clock frequency can be up to 64 MHz. The Flash access time for read access is minimum, write and erase operations are possible.

- **Range 2: Low-power range**
  
The main regulator provides a typical output voltage at 1.0 V. The system clock frequency can be up to 16 MHz. The Flash memory access time for a read access is increased as compared to Range 1; write and erase operations are not possible.

The voltage scaling is selected through the VOS bit in the PWR\_CR1 register.

The sequence to go from Range 1 to Range 2 is:

1. Reduce the system frequency to a value lower than 16 MHz
2. Adjust number of wait states according new frequency target in Range 2 (LATENCY bits in the FLASH\_ACR).
3. Program the VOS[1:0] bits to 10 in the **Power control register 1 (PWR\_CR1)**.

The sequence to go from Range 2 to Range 1 is:

1. Program the VOS[1:0] bits to 01 in the **Power control register 1 (PWR\_CR1)**.
2. Wait until the VOSF flag is cleared in the **Power status register 2 (PWR\_SR2)**.
3. Adjust number of wait states according new frequency target in Range 1 (LATENCY bits in the **FLASH access control register (FLASH\_ACR)**).
4. Increase the system frequency.
4.2 Power supply supervisor

4.2.1 Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR)

The device features an integrated power-on reset (POR) / power-down reset (PDR), coupled with a brown-out reset (BOR) circuitry. The POR/PDR is active in all power modes. The BOR can be enabled or disabled only through option bytes. It is not available in Shutdown mode.

When the BOR is enabled, four BOR levels can be selected through option bytes, with independent configuration for rising and falling thresholds. During power-on, the BOR keeps the device under reset until the $V_{DD}$ supply voltage reaches the specified BOR rising threshold ($V_{BORRX}$). At this point, the device reset is released and the system can start. During power-down, when $V_{DD}$ drops below the selected BOR falling threshold ($V_{BORFX}$), the device is put under reset again.

Warning: It is not allowed to configure BOR falling threshold ($V_{BORFX}$) to a value higher than BOR rising threshold ($V_{BORRX}$).

---

Figure 6. POR, PDR, and BOR thresholds

1. The reset temporization $t_{RSTTEMPO}$ starts when $V_{DD}$ crosses $V_{POR}$ threshold, indifferently from the configuration of the BOR Option bits.
For more details on the brown-out reset thresholds, refer to the electrical characteristics section in the datasheet.

### 4.2.2 Programmable voltage detector (PVD)

The PVD can be used to monitor the $V_{DD}$ power supply by comparing it to the thresholds selected through PVDRT[2:0] bits (rising thresholds) and PVDFT[2:0] bits (falling thresholds) in the Power control register 2 (PWR_CR2). $V_{PVDFX}$ should always be set to a lower voltage level than $V_{PVDFX}$.

The PVD is enabled by setting the PVDE bit.

A PVDO flag is available in the Power status register 2 (PWR_SR2). It indicates if $V_{DD}$ is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if enabled through the EXTI registers. The PVD output interrupt can be generated when $V_{DD}$ drops below the PVD threshold and/or when $V_{DD}$ rises above the PVD threshold depending on EXTI line16 rising/falling edge configuration. As an example, the service routine could perform emergency shutdown tasks.

#### Figure 7. PVD thresholds

![PVD thresholds diagram](image-url)
4.3 Low-power modes

By default, the microcontroller is in Run mode after a system or a power Reset. Several low-power modes are available to save power when the CPU does not need to be kept running, for example when waiting for an external event. It is up to the user to select the mode that gives the best compromise between low-power consumption, short startup time and available wakeup sources.

The device features seven low-power modes:

- **Sleep mode**: CPU clock off, all peripherals including Cortex®-M0+ core peripherals such as NVIC, SysTick, etc. can run and wake up the CPU when an interrupt or an event occurs. Refer to Section 4.3.4: Sleep mode.

- **Low-power run mode**: This mode is achieved when the system clock frequency is reduced below 2 MHz. The code is executed from the SRAM or the Flash memory. The regulator is in low-power mode to minimize the regulator's operating current. Refer to Section 4.3.2: Low-power run mode (LP run).

- **Low-power sleep mode**: This mode is entered from the Low-power run mode: Cortex®-M0+ is off. Refer to Section 4.3.5: Low-power sleep mode (LP sleep).

- **Stop 0 and Stop 1 modes**: SRAM and all registers content are retained. All clocks in the V_CORE domain are stopped, the PLL, the HSI16 and the HSE are disabled. The LSI and the LSE can be kept running. The RTC and TAMP can remain active (Stop mode with RTC, Stop mode without RTC). Some peripherals with the wakeup capability can enable the HSI16 RC during the Stop mode to detect their wakeup condition.

  In Stop 0 mode, the main regulator remains ON, which allows the fastest wakeup time but with higher consumption. The active peripherals and wakeup sources are the same as in Stop 1 mode.

  The system clock, when exiting Stop 0 or Stop 1 mode, is the HSISYS clock. If the device is configured to wake up in Low-power run mode, the HSIDIV bits in RCC_CR register must be configured prior to entering Stop mode to provide a frequency not greater than 2 MHz.

  Refer to Section 4.3.6: Stop 0 mode for details on Stop 0 mode.

- **Standby mode**: V_CORE domain is powered off. However, it is possible to preserve SRAM content:
  - Standby mode with SRAM retention when the bit RRS is set in PWR_CR3 register. In this case, SRAM is supplied by the low-power regulator.
  - Standby mode when the bit RRS is cleared in PWR_CR3 register. In this case the main regulator and the low-power regulator are powered off.

  All clocks in the V_CORE domain are stopped and the PLL, the HSI16, and the HSE oscillators are disabled. The LSI and the LSE oscillators can be kept running. The RTC can remain active (Standby mode with RTC, Standby mode without RTC). The system clock, when exiting Standby modes, is the HSI16 oscillator clock.

  Refer to Section 4.3.8: Standby mode.

- **Shutdown mode**: V_CORE domain is powered off. All clocks in the V_CORE domain are stopped, the PLL, the HSI16, the LSI and the HSE oscillators are disabled. The LSE can be kept running. The system clock, when exiting Shutdown mode, is the HSI16 oscillator clock. In this mode, the supply voltage monitoring is disabled and the product...
behavior is not guaranteed in case of a power voltage drop. Refer to Section 4.3.9: Shutdown mode.

In addition, the power consumption in Run mode can be reduced by one of the following means:
- Slowing down the system clocks
- Gating the clocks to the APB and AHB peripherals when they are unused.

Figure 8. Low-power modes state diagram
### Table 20. Low-power mode summary

<table>
<thead>
<tr>
<th>Mode name</th>
<th>Entry</th>
<th>Wakeup source$^{(1)}$</th>
<th>Wakeup system clock</th>
<th>Effect on clocks</th>
<th>Voltage regulators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep (Sleep-now or Sleep-on-exit)</td>
<td>WFI or Return from ISR</td>
<td>Any interrupt</td>
<td>Same as before entering Sleep mode</td>
<td>CPU clock OFF no effect on other clocks or analog clock sources</td>
<td>MR OFF LPR ON</td>
</tr>
<tr>
<td></td>
<td>WFE</td>
<td>Wakeup event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low-power run</td>
<td>Set LPR bit</td>
<td>Clear LPR bit</td>
<td>No change</td>
<td>None</td>
<td>OFF MR OFF LPR ON</td>
</tr>
<tr>
<td>Low-power sleep</td>
<td>Set LPR bit + WFI or Return from ISR</td>
<td>Any interrupt</td>
<td>Same as before entering Low-power sleep mode</td>
<td>CPU clock OFF no effect on other clocks or analog clock sources</td>
<td>MR OFF OFF LPR ON</td>
</tr>
<tr>
<td></td>
<td>Set LPR bit + WFE</td>
<td>Wakeup event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop 0</td>
<td>LPMS=“000” + SLEEPDEEP bit + WFI or Return from ISR or WFE</td>
<td>Any EXTI line (configured in the EXTI registers)</td>
<td></td>
<td></td>
<td>MR OFF ON MR OFF</td>
</tr>
<tr>
<td>Stop 1</td>
<td>LPMS=“001” + SLEEPDEEP bit + WFI or Return from ISR or WFE</td>
<td>Specific peripherals events</td>
<td></td>
<td></td>
<td>MR OFF ON MR OFF</td>
</tr>
<tr>
<td>Standby with SRAM</td>
<td>LPMS=“011”+ Set RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE</td>
<td>WKUP pin edge, RTC event, TAMPER event, external reset on NRST pin, IWDG reset</td>
<td>HSISYS</td>
<td></td>
<td>MR OFF OFF MO ON</td>
</tr>
<tr>
<td>Standby</td>
<td>LPMS=“011” + Clear RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE</td>
<td></td>
<td></td>
<td></td>
<td>MR OFF OFF OFF ON</td>
</tr>
<tr>
<td>Shutdown</td>
<td>LPMS=“1--” + SLEEPDEEP bit + WFI or Return from ISR or WFE</td>
<td>WKUP pin edge, RTC event, TAMPER event, external reset on NRST pin</td>
<td></td>
<td></td>
<td>MR OFF OFF OFF LSE</td>
</tr>
</tbody>
</table>

1. Refer to Table 21: Functionalities depending on the working mode.
<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Run</th>
<th>Sleep</th>
<th>Low-power run</th>
<th>Low-power sleep</th>
<th>Stop 0/1 Wakeup capability</th>
<th>Standby Wakeup capability</th>
<th>Shutdown Wakeup capability</th>
<th>VBAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Y</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Flash memory</td>
<td>Y</td>
<td>Y</td>
<td>O(^{(2)})</td>
<td>O(^{(2)})</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SRAM</td>
<td>Y</td>
<td>Y(^{(3)})</td>
<td>Y</td>
<td>Y(^{(3)})</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>Backup Registers</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Brown-out reset (BOR)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>Programmable Voltage Detector (PVD)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>DMA</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Oscillator HSI16</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(^{(5)})</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>High Speed External (HSE)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Low Speed Internal (LSI)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>Low Speed External (LSE)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>PLL</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Clock Security System (CSS)</td>
<td>O</td>
<td>O</td>
<td>O(^{(6)})</td>
<td>O(^{(6)})</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Clock Security System on LSE</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>RTC / Auto wakeup</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>TAMPx (x=1,2)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>USARTx (x=1,2)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(^{(7)})</td>
<td>O(^{(7)})</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>USARTx (x=3,4)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(^{(7)})</td>
<td>O(^{(7)})</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Low-power UART (LPUART1)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(^{(7)})</td>
<td>O(^{(7)})</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I2C1</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(^{(8)})</td>
<td>O(^{(8)})</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I2C2</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(^{(8)})</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SPIx (x=1,2)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(^{(8)})</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ADC</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(^{(8)})</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DAC</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(^{(8)})</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VREFBUF</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(^{(8)})</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>COMPx (x=1,2)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(^{(8)})</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 21. Functionalities depending on the working mode\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Run</th>
<th>Sleep</th>
<th>Low-power run</th>
<th>Low-power sleep</th>
<th>Stop 0/1 Wakeup capability</th>
<th>Standby Wakeup capability</th>
<th>Shutdown Wakeup capability</th>
<th>VBAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature sensor</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Timers (TIMx)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Low-power timer 1 (LPTIM1)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>Low-power timer 2 (LPTIM2)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>Independent watchdog (IWDG)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>Window watchdog (WWDG)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SysTick timer</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Random number generator (RNG)(^{(9)})</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AES hardware accelerator(^{(9)})</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CRC calculation unit</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GPIOs</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
</tbody>
</table>

1. Legend: Y = Yes (Enable). O = Optional (Disable by default. Can be enabled by software). - = Not available.
2. The Flash can be configured in power-down mode. By default, it is not in power-down mode.
3. The SRAM clock can be gated ON or OFF.
4. SRAM content is preserved when the bit RRS is set in PWR_CR3 register.
5. Some peripherals with wakeup from Stop capability can request HSI16 to be enabled. In this case, HSI16 is woken up by the peripheral, and only feeds the peripheral which requested it. HSI16 is automatically put OFF when the peripheral does not need it anymore.
6. If CSS is used on HSE clock in Low power run or Low power sleep modes, configure HSIDIV such as not to drive SYSCLK clock above the maximum frequency for either mode, in case of external clock failure detection.
7. USART and LPUART reception is functional in Stop mode, and generates a wakeup interrupt on Start, address match or received frame event.
8. I2C address detection is functional in Stop mode, and generates a wakeup interrupt in case of address match.
10. I/Os can be configured with internal pull-up, pull-down or floating in Standby mode.
11. I/Os with wakeup from Standby/Shutdown mode capability (WKUPx).
12. I/Os can be configured with internal pull-up, pull-down or floating in Shutdown mode but the configuration is lost when exiting Shutdown mode.
Debug mode

By default, the debug connection is lost if the user application puts the MCU in Stop 0, Stop1, Shutdown, or Standby mode while the debug features are used. This is due to the fact that the Cortex®-M0+ core is no longer clocked.

However, by setting some configuration bits in the DBGMCU_CR register, the software can be debugged even when using the low-power modes extensively. For more details, refer to Section 37.9.1: Debug support for low-power modes.

4.3.1 Run mode

Slowing down system clocks

In Run mode, the speed of the system clocks (SYSCLK, HCLK, PCLK) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down the peripherals before entering Sleep mode.

For more details, refer to Section 5.4.3: Clock configuration register (RCC_CFGR).

Peripheral clock gating

In Run mode, the HCLK and PCLK for individual peripherals and memories can be stopped at any time to reduce the power consumption.

To further reduce the power consumption in Sleep mode, the peripheral clocks can be disabled prior to executing the WFI or WFE instructions.

The peripheral clock gating is controlled by the RCC_AHBENR and RCC_APBENRx registers.

Disabling the peripherals clocks in Sleep mode can be performed automatically by resetting the corresponding bit in the RCC_AHBSMENR and RCC_APBSMENRx registers.

4.3.2 Low-power run mode (LP run)

To further reduce the consumption when the system is in Run mode, the regulator can be configured in low-power mode. In this mode, the system frequency should not exceed 2 MHz.

Please refer to the product datasheet for more details on voltage regulator and peripherals operating conditions.

I/O states in Low-power run mode

In Low-power run mode, all I/O pins keep the same state as in Run mode.

Entering Low-power run mode

To enter Low-power run mode, proceed as follows:

1. Optional: Jump into the SRAM and power-down the Flash memory by setting the FPD_LPRUN bit in the Power control register 1 (PWR_CR1).
2. Decrease the system clock frequency below 2 MHz.
3. Force the regulator in low-power mode by setting the LPR bit in the PWR_CR1 register.

Refer to Table 22: Low-power run on how to enter Low-power run mode.
Exiting Low-power run mode

To exit Low-power run mode, proceed as follows:

1. Force the regulator in main mode by clearing the LPR bit in the Power control register 1 (PWR_CR1).
2. Wait until REGLPF bit is cleared in the Power status register 2 (PWR_SR2).
3. Increase the system clock frequency.

Refer to Table 22: Low-power run on how to exit Low-power run mode.

<table>
<thead>
<tr>
<th>Low-power run mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode entry</td>
<td>Decrease the system clock frequency below 2 MHz</td>
</tr>
<tr>
<td></td>
<td>LPR = 1</td>
</tr>
<tr>
<td>Mode exit</td>
<td>LPR = 0</td>
</tr>
<tr>
<td></td>
<td>Wait until REGLPF = 0</td>
</tr>
<tr>
<td></td>
<td>Increase the system clock frequency</td>
</tr>
<tr>
<td>Wakeup latency</td>
<td>Regulator wakeup time from low-power mode</td>
</tr>
</tbody>
</table>

4.3.3 Low-power modes

Entering low-power modes

The MCU enters low-power modes by executing the WFI (wait for interrupt), or WFE (wait for event) instructions, or when the SLEEPONEXIT bit in the Cortex®-M0+ system control register is set on return from ISR.

Entering low-power mode through WFI or WFE will be executed only if no interrupt is pending or no event is pending.

Exiting low-power modes

The MCU exits Sleep and Stop low-power modes in a way depending on how the low-power mode was entered:

- If the WFI instruction or Return from ISR was used to enter low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the device.
- If the WFE instruction is used to enter low-power mode, the MCU exits low-power mode as soon as an event occurs. The wakeup event can be generated either by:
  - NVIC IRQ interrupt.
    When SEVONPEND = 0 in the Cortex®-M0+ system control register: by enabling an interrupt in the peripheral control register and in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
When SEVONPEND = 1 in the Cortex®-M0+ system control register: by enabling an interrupt in the peripheral control register and optionally in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and when enabled the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.

All NVIC interrupts will wakeup the MCU, even the disabled ones.

Event

Configuring a EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bits corresponding to the event line is not set.

It may be necessary to clear the interrupt flag in the peripheral.

The MCU exits Standby and Shutdown low-power modes upon an external reset (NRST pin), an IWDG reset, a rising or falling edge on one of enabled WKUPx pins, or upon an RTC event. See Figure 282: RTC block diagram.

After waking up from Standby or Shutdown mode, program execution restarts in the same way as after a reset (boot pin sampling, option bytes loading, reset vector is fetched, etc.).

### 4.3.4 Sleep mode

**I/O states in Sleep mode**

In Sleep mode, all I/O pins keep the same state as in Run mode.

**Entering Sleep mode**

The MCU enters Sleep mode according to section Entering low-power modes, when the SLEEPDEEP bit in the Cortex®-M0+ System Control register is clear.

Refer to Table 23: Sleep mode summary for details on how to enter Sleep mode.

**Exiting Sleep mode**

The MCU exits Sleep mode according to Exiting low-power modes.

Refer to Table 23: Sleep mode summary for more details on how to exit Sleep mode.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode entry</td>
<td>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP = 0</td>
</tr>
<tr>
<td></td>
<td>– No interrupt (for WFI) or event (for WFE) is pending</td>
</tr>
<tr>
<td></td>
<td>Refer to the Cortex®-M0+ system control register.</td>
</tr>
<tr>
<td></td>
<td>On return from ISR while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP = 0 and</td>
</tr>
<tr>
<td></td>
<td>– SLEEPONEXIT = 1</td>
</tr>
<tr>
<td></td>
<td>– No interrupt is pending</td>
</tr>
<tr>
<td></td>
<td>Refer to the Cortex®-M0+ system control register.</td>
</tr>
</tbody>
</table>
4.3.5 Low-power sleep mode (LP sleep)

Please refer to the product datasheet for more details on voltage regulator and peripherals operating conditions.

I/O states in Low-power sleep mode

In Low-power sleep mode, all I/O pins keep the same state as in Run mode.

Entering Low-power sleep mode

The MCU enters Low-power sleep mode from Low-power run mode according to Entering low-power modes, when the SLEEPDEEP bit in the Cortex®-M0+ System Control register is clear.

Refer to Table 24: Low-power sleep mode summary for details on how to enter Low-power sleep mode.

Exiting Low-power sleep mode

The MCU exits Low-power sleep mode according to Exiting low-power modes. When exiting Low-power sleep mode by issuing an interrupt or an event, the MCU is in Low-power run mode.

Refer to Table 24: Low-power sleep mode summary for details on how to exit Low-power sleep mode.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode exit</td>
<td>If WFI or return from ISR was used for entry</td>
</tr>
<tr>
<td></td>
<td>Interrupt: refer to Table 48: Vector table</td>
</tr>
<tr>
<td></td>
<td>If WFE was used for entry and SEVONPEND = 0:</td>
</tr>
<tr>
<td></td>
<td>Wakeup event: refer to Section 12.3.2: EXTI direct event input wakeup</td>
</tr>
<tr>
<td></td>
<td>If WFE was used for entry and SEVONPEND = 1:</td>
</tr>
<tr>
<td></td>
<td>Interrupt even when disabled in NVIC: refer to Table 48: Vector table or</td>
</tr>
<tr>
<td></td>
<td>Wakeup event: refer to Section 12.3.2: EXTI direct event input wakeup</td>
</tr>
<tr>
<td>Wakeup latency</td>
<td>None</td>
</tr>
</tbody>
</table>
4.3.6 Stop 0 mode

The Stop 0 mode is based on the Cortex®-M0+ deepsleep mode combined with the peripheral clock gating. The voltage regulator is configured in main regulator mode. In Stop 0 mode, all clocks in the VCORE domain are stopped; the PLL, the HSI16 and the HSE oscillators are disabled. Some peripherals with the wakeup capability (I2C1, USART1, USART2, and LPUART1) can switch on the HSI16 to receive a frame, and switch off the HSI16 after receiving the frame if it is not a wakeup frame. In this case, the HSI16 clock is propagated only to the peripheral requesting it.

SRAM and register contents are preserved.

The BOR is available in Stop 0 mode.

The BOR and PDR can be activated to sample periodically the supply voltage. This option enabled by setting the ULPEN bit of the PWR_CR3 register allows decreasing the current consumption in this mode, but any drop of the voltage below the operating conditions between two active periods of the supply detector results in a non-generation of PDR reset.

I/O states in Stop 0 mode

In the Stop 0 mode, all I/O pins keep the same state as in the Run mode.

---

Table 24. Low-power sleep mode summary

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode entry</td>
<td>Low-power sleep mode is entered from the Low-power run mode. WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 – No interrupt (for WFI) or event (for WFE) is pending Refer to the Cortex®-M0+ System Control register. On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt is pending Refer to the Cortex®-M0+ System Control register.</td>
</tr>
<tr>
<td>Mode exit</td>
<td>If WFI or Return from ISR was used for entry Interrupt: refer to Table 48: Vector table If WFE was used for entry and SEVONPEND = 0: Wakeup event: refer to Section 12.3.2: EXTI direct event input wakeup If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to Table 48: Vector table Wakeup event: refer to Section 12.3.2: EXTI direct event input wakeup After exiting Low-power sleep mode, the MCU is in Low-power run mode.</td>
</tr>
<tr>
<td>Wakeup latency</td>
<td>None</td>
</tr>
</tbody>
</table>
Entering Stop 0 mode

The MCU enters Stop 0 mode according to section *Entering low-power modes*, when the SLEEPDEEP bit in the Cortex®-M0+ System Control register is set. Refer to *Table 25: Stop 0 mode summary* for details on how to enter Stop 0 mode.

If Flash memory programming is ongoing, the Stop 0 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, the Stop 0 mode entry is delayed until the APB access is finished.

In Stop 0 mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started, it cannot be stopped except upon a reset. See *Section 27.3: IWDG functional description*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the *RTC domain control register (RCC_BDCR)*.
- Internal RC oscillator (LSI): this is configured by the LSION bit in the *Control/status register (RCC_CSR)*.
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the *RTC domain control register (RCC_BDCR)*.

Several peripherals can be used in Stop 0 mode and can add consumption if they are enabled and clocked by LSI or LSE, or when they request the HSI16 clock: LPTIM1, LPTIM2, USART1, USART2, LPUART, and I2C1.

The DAC, the comparators, and the PVD can be used in Stop 0 mode.

The ADC, the VREFBUF buffer, and the temperature sensor can consume power during the Stop 0 mode, unless they are disabled before entering this mode.

Exiting Stop 0 mode

The MCU exits Stop 0 mode according to section *Entering low-power modes*.

Refer to *Table 25: Stop 0 mode summary* for details on how to exit Stop 0 mode.

When exiting Stop 0 mode by issuing an interrupt or a wakeup event, the HSISYS oscillator is selected as system clock. If the device is configured to wake up in Low-power run mode, the HSIDIV bits in RCC_CR register must be configured prior to entering Stop 0 mode to provide a frequency not greater than 2 MHz.

When exiting Stop 0 mode, the MCU is either in Run mode (Range 1 or Range 2 depending on VOS bit in PWR_CR1) or in Low-power run mode if the bit LPR is set in the *Power control register 1 (PWR_CR1)*.
### Table 25. Stop 0 mode summary

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Mode entry** | WiFi (Wait for Interrupt) or WFE (Wait for Event) while:  
| | – SLEEPDEEP bit is set in Cortex®-M0+ System Control register  
| | – No interrupt (for WiFi) or event (for WFE) is pending  
| | – LPMS = “000” in PWR_CR1  
| | On Return from ISR while:  
| | – SLEEPDEEP bit is set in Cortex®-M0+ System Control register  
| | – SLEEPONEXIT = 1  
| | – No interrupt is pending  
| | – LPMS = “000” in PWR_CR1  
| **Note:** | To enter Stop 0 mode, all EXTI Line pending bits (in EXTI rising edge pending register (EXTI_RPR1) and EXTI falling edge pending register (EXTI_FPR1)), and the peripheral flags generating wakeup interrupts must be cleared. Otherwise, the Stop 0 mode entry procedure is ignored and program execution continues. |
| **Mode exit** | If WiFi or Return from ISR was used for entry  
| | Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to Table 48: Vector table.  
| | If WFE was used for entry and SEVONPEND = 0:  
| | Any EXTI Line configured in event mode. Refer to Section 12.3.2: EXTI direct event input wakeup.  
| | If WFE was used for entry and SEVONPEND = 1:  
| | Any EXTI Line configured in Interrupt mode (even if the corresponding EXTI Interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to Table 48: Vector table.  
| | Wakeup event: refer to Section 12.3.2: EXTI direct event input wakeup.  
| **Wakeup latency** | Longest wakeup time between HSI16 wakeup time and Flash wakeup time from Stop 0 mode. |
4.3.7 Stop 1 mode

The Stop 1 mode is the same as Stop 0 mode except that the main regulator is off, and only the low-power regulator is on. Stop 1 mode can be entered from Run mode and from Low-power run mode.

Refer to Table 26: Stop 1 mode summary for details on how to enter and exit Stop 1 mode.

Table 26. Stop 1 mode summary

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode entry</strong></td>
<td>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP bit is set in Cortex®-M0+ System Control register</td>
</tr>
<tr>
<td></td>
<td>– No interrupt (for WFI) or event (for WFE) is pending</td>
</tr>
<tr>
<td></td>
<td>– LPMS = “001” in PWR_CR1</td>
</tr>
<tr>
<td></td>
<td>On Return from ISR while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP bit is set in Cortex®-M0+ System Control register</td>
</tr>
<tr>
<td></td>
<td>– SLEEPONEXIT = 1</td>
</tr>
<tr>
<td></td>
<td>– No interrupt is pending</td>
</tr>
<tr>
<td></td>
<td>– LPMS = “001” in PWR_CR1</td>
</tr>
<tr>
<td><strong>Note:</strong> To enter Stop 1 mode, all EXTI Line pending bits (in EXTI rising edge pending register (EXTI_RPR1) and EXTI falling edge pending register (EXTI_FPR1)), and the peripheral flags generating wakeup interrupts must be cleared. Otherwise, the Stop 1 mode entry procedure is ignored and program execution continues.</td>
<td></td>
</tr>
<tr>
<td><strong>Mode exit</strong></td>
<td>If WFI or Return from ISR was used for entry</td>
</tr>
<tr>
<td></td>
<td>Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to Table 48: Vector table.</td>
</tr>
<tr>
<td></td>
<td>If WFE was used for entry and SEVONPEND = 0:</td>
</tr>
<tr>
<td></td>
<td>Any EXTI Line configured in event mode. Refer to Section 12.3.2: EXTI direct event input wakeup.</td>
</tr>
<tr>
<td></td>
<td>If WFE was used for entry and SEVONPEND = 1:</td>
</tr>
<tr>
<td></td>
<td>Any EXTI Line configured in Interrupt mode (even if the corresponding EXTI Interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to Table 48: Vector table.</td>
</tr>
<tr>
<td></td>
<td>Wakeup event: refer to Section 12.3.2: EXTI direct event input wakeup</td>
</tr>
<tr>
<td><strong>Wakeup latency</strong></td>
<td>Longest wakeup time between HSI16 wakeup time and regulator wakeup time from Low-power mode + Flash wakeup time from Stop 1 mode.</td>
</tr>
</tbody>
</table>
4.3.8 Standby mode

The Standby mode allows to achieve the lowest power consumption with BOR. It is based on the Cortex®-M0+ deepsleep mode, with the voltage regulators disabled (except when the SRAM content is preserved). The PLL, the HSI16 and the HSE oscillators are also switched off.

The content of the registers is lost except for the registers in the RTC domain and Standby circuitry (see Figure 5). The SRAM content is lost except if the RRS bit is set in the PWR_CR3 register. In this case the low-power regulator is on and provides the supply to the SRAM only.

The BOR is available in Standby mode.

The BOR and PDR can be activated to sample periodically the supply voltage. This option enabled by setting the ULPEN bit of the PWR_CR3 register allows to decrease the current consumption in this mode, but any drop of the voltage below the operating conditions between two active periods of the supply detector results in a non-generation of PDR reset.

I/O states in Standby mode

In the Standby mode, the I/Os can be configured either with a pull-up (refer to PWR_PUCRx registers (x=A, B, C, D, F), or with a pull-down (refer to PWR_PDCRx registers (x=A, B, C, D, F)), or can be kept in analog mode.

The RTC outputs on PC13 and PA4 are functional in Standby mode. PC14 and PC15 used for LSE are also functional. Five wakeup pins (WKUPx, x=1,2,4,5,6) and the two tampers are available.

Entering Standby mode

The MCU enters Standby mode according to Entering low-power modes, when the SLEEPDEEP bit in the Cortex®-M0+ System Control register is set.

Refer to Table 27: Standby mode summary for details on how to enter Standby mode.

In Standby mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a reset. See Section 27.3: IWDG functional description.
- Real-time clock (RTC) and tamper (TAMP): this is configured by the RTCEN bit in the RTC domain control register (RCC_BDCR)
- Internal RC oscillator (LSI): this is configured by the LSION bit in the Control/status register (RCC_CSR).
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the RTC domain control register (RCC_BDCR)
Exiting Standby mode

The MCU exits Standby mode according to section *Entering low-power modes*. The SBF status flag in the *Power control register 3 (PWR_CR3)* indicates that the MCU was in Standby mode. All registers are reset after wakeup from Standby except for *Power control register 3 (PWR_CR3)*.

Refer to *Table 27: Standby mode summary* for more details on how to exit Standby mode.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode entry</strong></td>
<td>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP bit is set in Cortex®-M0+ System Control register</td>
</tr>
<tr>
<td></td>
<td>– No interrupt (for WFI) or event (for WFE) is pending</td>
</tr>
<tr>
<td></td>
<td>– LPMS = “011” in <em>Power control register 1 (PWR_CR1)</em></td>
</tr>
<tr>
<td></td>
<td>– WUFx bits are cleared in <em>Power status register 1 (PWR_SR1)</em></td>
</tr>
<tr>
<td></td>
<td>On return from ISR while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP bit is set in Cortex®-M0+ System Control register</td>
</tr>
<tr>
<td></td>
<td>– SLEEPONEXIT = 1</td>
</tr>
<tr>
<td></td>
<td>– No interrupt is pending</td>
</tr>
<tr>
<td></td>
<td>– LPMS = “011” in <em>Power control register 1 (PWR_CR1)</em></td>
</tr>
<tr>
<td></td>
<td>– WUFx bits are cleared in <em>Power status register 1 (PWR_SR1)</em></td>
</tr>
<tr>
<td></td>
<td>– The RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup, tamper or timestamp flags) is cleared</td>
</tr>
<tr>
<td><strong>Mode exit</strong></td>
<td>WKUPx pin edge, RTC event, TAMP event, external reset on NRST pin, IWDG reset, BOR</td>
</tr>
<tr>
<td><strong>Wakeup latency</strong></td>
<td>Reset phase</td>
</tr>
</tbody>
</table>
4.3.9 Shutdown mode

The Shutdown mode allows to achieve the lowest power consumption. It is based on the deepsleep mode, with the voltage regulator disabled. The V\text{CORE} domain is consequently powered off. The PLL, the HSI16, the LSI and the HSE oscillators are also switched off.

SRAM and register contents are lost except for registers in the RTC domain. The POR/PDR and BOR are not available in Shutdown mode. No power voltage monitoring is possible in this mode. As a result, the switch of the RTC domain to V\text{BAT} supply when V\text{DD} supply is lost is not supported.

I/O states in Shutdown mode

In the Shutdown mode, the I/Os can be configured either with a pull-up (refer to PWR\_PUCRx registers (x=A, B, C, D, F)), or with a pull-down (refer to PWR\_PDCRx registers (x=A, B, C, D, F)), or can be kept in analog state. However this configuration is lost when exiting Shutdown mode due to the power-on reset.

The RTC outputs on PC13 are functional in Shutdown mode. PC14 and PC15 used for LSE are also functional. Five wakeup pins (WKUPx, x=1,2,4,5,6) and the two tampers are available.

Entering Shutdown mode

The MCU enters Shutdown mode according to section *Entering low-power modes*, when the SLEEPDEEP bit in the Cortex®-M0+ System Control register is set.

Refer to *Table 28: Shutdown mode summary* for details on how to enter Shutdown mode.

In Shutdown mode, the following features can be selected by programming individual control bits:

- Real-time clock (RTC) and tamper (TAMP): this is configured by the RTCEN bit in the RTC domain control register (RCC\_BDCR).
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the RTC domain control register (RCC\_BDCR).

**Caution:** The RTC domain content is lost when V\text{DD} is powered down under Shutdown mode.
Exiting Shutdown mode

The MCU exits Shutdown mode according to section *Exiting low-power modes*. A power-on reset occurs when exiting from Shutdown mode. All registers (except for the ones in the RTC domain) are reset after wakeup from Shutdown.

Refer to *Table 28: Shutdown mode summary* for more details on how to exit Shutdown mode.

**Table 28. Shutdown mode summary**

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode entry</td>
<td>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP bit is set in Cortex®-M0+ system control register</td>
</tr>
<tr>
<td></td>
<td>– No interrupt (for WFI) or event (for WFE) is pending</td>
</tr>
<tr>
<td></td>
<td>– LPMS[2:0] = 1XX in Power control register 1 (PWR_CR1)</td>
</tr>
<tr>
<td></td>
<td>– WUFx bits are cleared in Power status register 1 (PWR_SR1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mode exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKUPx pin edge, RTC event, external Reset on NRST pin</td>
</tr>
</tbody>
</table>

| Wakeup latency | Reset phase |

### 4.3.10 Auto-wakeup from low-power mode

The RTC can be used to wakeup the MCU from low-power mode without depending on an external interrupt (Auto-wakeup mode). The RTC provides a programmable time base for waking up from Stop (0, 1), Shutdown or Standby modes at regular intervals. For this purpose, two of the three alternative RTC clock sources can be selected by programming the RTCSEL[1:0] bits in the RTC domain control register (RCC_BDCR):

- **Low-power 32.768 kHz external crystal oscillator (LSE OSC)**
  - This clock source provides a precise time base with very low-power consumption.

- **Low-power internal RC Oscillator (LSI)**
  - This clock source has the advantage of saving the cost of the 32.768 kHz crystal. This internal RC Oscillator is designed to add minimum power consumption.

To wake up from Stop mode with an RTC alarm or an RTC wakeup event, it is necessary to:

- Configure the EXTI Line 19 to be sensitive to rising edge.
- Configure the RTC to generate the wakeup event.

To wake up from Standby or Shutdown mode, there is no need to configure the EXTI line 19.
## 4.4 PWR registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

### 4.4.1 Power control register 1 (PWR_CR1)

Address offset: 0x00

Reset value: 0x0000 0208. This register is reset after wakeup from Standby mode.

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

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **LPR**: Low-power run

When this bit is set, the regulator is switched from main mode (MR) to low-power mode (LPR).

Bits 13:11 Reserved, must be kept at reset value.

Bits 10:9 **VOS**: Voltage scaling range selection

- 00: Cannot be written (forbidden by hardware)
- 01: Range 1
- 10: Range 2
- 11: Cannot be written (forbidden by hardware)

Bit 8 **DBP**: Disable RTC domain write protection

In reset state, the RTC and backup registers are protected against parasitic write access. This bit must be set to enable write access to these registers.

- 0: Access to RTC and backup registers disabled
- 1: Access to RTC and backup registers enabled

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **FPD_LPSLP**: Flash memory powered down during Low-power sleep mode

This bit determines whether the Flash memory is put in power-down mode or remains in idle mode when the device enters Low-power sleep mode.

- 0: Flash memory idle
- 1: Flash memory powered down
4.4.2 Power control register 2 (PWR_CR2)

Address offset: 0x04

Reset value: 0x0000 0000. This register is reset when exiting Standby mode.

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

PVDRT[2:0]  PVDFT[2:0]  PVDE

rw  rw  rw  rw  rw  rw  rw  rw

Note: In Standby mode, SRAM contents are retained or not, depending on RRS bit setting in PWR_CR3.
Bits 31:7  Reserved, must be kept at reset value.

Bits 6:4  **PVDRT[2:0]**: Power voltage detector rising threshold selection.

These bits select the PVD rising threshold:
- 000: \( V_{PVDR0} \) (around 2.1 V)
- 001: \( V_{PVDR1} \) (around 2.2 V)
- 010: \( V_{PVDR2} \) (around 2.5 V)
- 011: \( V_{PVDR3} \) (around 2.6 V)
- 100: \( V_{PVDR4} \) (around 2.7 V)
- 101: \( V_{PVDR5} \) (around 2.9 V)
- 110: \( V_{PVDR6} \) (around 3.0 V)
- 111: PVD_IN pin voltage

*Note:* If this bitfield is set to 111, the voltage on PVD_IN pin is internally compared with \( V_{REFINT} \) for both rising and falling threshold and the PVDFT[2:0] bitfield has no effect.

*Note:* These bits are write-protected when the PVD_LOCK bit is set in the SYSCFG_CFRGR2 register. The protection can be reset only by a system reset.

Bits 3:1  **PVDFT[2:0]**: Power voltage detector falling threshold selection.

These bits select the PVD falling threshold:
- 000: \( V_{PVDF0} \) (around 2.0 V)
- 001: \( V_{PVDF1} \) (around 2.2 V)
- 010: \( V_{PVDF2} \) (around 2.4 V)
- 011: \( V_{PVDF3} \) (around 2.5 V)
- 100: \( V_{PVDF4} \) (around 2.6 V)
- 101: \( V_{PVDF5} \) (around 2.8 V)
- 110: \( V_{PVDF6} \) (around 2.9 V)
- 111: Not used

*Note:* The setting of this bitfield is ignored as long as the bitfield PVDRT[2:0] is set to 111.

*Note:* These bits are write-protected when the PVD_LOCK bit is set in the SYSCFG_CFRGR2 register. The protection can be reset only by a system reset.

Bit 0  **PVDE**: Power voltage detector enable

0: Power voltage detector disable.
1: Power voltage detector enable.

*Note:* This bit is write-protected when the PVD_LOCK bit is set in the SYSCFG_CFRGR2 register. The protection can be reset only by a system reset.
4.4.3 Power control register 3 (PWR_CR3)

Address offset: 0x08

Reset value: 0x0000 8000. This register is not reset when exiting Standby modes and with the PWRRST bit in the APB peripheral reset register 1 (RCC_APBRSTR1).

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EIWUL**: Enable internal wakeup line

0: Disable
1: Enable

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **APC**: Apply pull-up and pull-down configuration

This bit determines whether the I/O pull-up and pull-down configurations defined in the PWR_PUCRx and PWR_PDCRx registers are applied.

0: Not applied
1: Applied

Bit 9 **ULPEN**: Ultra-low-power enable

Enable/disable periodical sampling of supply voltage in Stop and Standby modes for detecting condition of PDR and BOR reset.

0: Disable (the supply voltage is monitored continuously)
1: Enable

When set, the supply voltage is sampled for PDR/BOR reset condition only periodically and not continuously, in order to save power.

**Caution:** When enabled, and if the supply voltage drops below the minimum operating condition between two supply voltage samples, the reset condition is missed and no reset is generated.

Bit 8 **RRS**: SRAM retention in Standby mode

0: SRAM is powered off in Standby mode (SRAM content is lost).
1: SRAM is powered by the low-power regulator in Standby mode (SRAM content is kept).

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **EWUP6**: Enable WKUP6 wakeup pin

When this bit is set, the WKUP6 external wakeup pin is enabled and triggers a wakeup from Standby or Shutdown mode when a rising or a falling edge occurs. The active edge is configured through WP6 bit in the PWR_CR4 register.
Power control (PWR) RM0444

4.4.4 Power control register 4 (PWR_CR4)

Address offset: 0x0C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with the PWRRST bit in the APB peripheral reset register 1 (RCC_APBRSTR1).

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

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

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

Bit 9 VBR5: VBAT battery charging resistor selection
0: Charge VBAT through a 5 kOhms resistor
1: Charge VBAT through a 1.5 kOhms resistor

Bit 8 VBE: VBAT battery charging enable
0: VBAT battery charging disable
1: VBAT battery charging enable

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 WP6: WKUP6 wakeup pin polarity
This bit defines the polarity used for an event detection on WKUP6 external wakeup pin:
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)
Bit 4 **WP5**: WKUP5 wakeup pin polarity  
This bit defines the polarity used for an event detection on WKUP5 external wakeup pin:  
0: Detection on high level (rising edge)  
1: Detection on low level (falling edge)  

Bit 3 **WP4**: WKUP4 wakeup pin polarity  
This bit defines the polarity used for an event detection on WKUP4 external wakeup pin:  
0: Detection on high level (rising edge)  
1: Detection on low level (falling edge)  

Bit 2 Reserved, must be kept at reset value.  

Bit 1 **WP2**: WKUP2 wakeup pin polarity  
This bit defines the polarity used for an event detection on WKUP2 external wakeup pin:  
0: Detection on high level (rising edge)  
1: Detection on low level (falling edge)  

Bit 0 **WP1**: WKUP1 wakeup pin polarity  
This bit defines the polarity used for an event detection on WKUP1 external wake-up pin:  
0: Detection on high level (rising edge)  
1: Detection on low level (falling edge)  

### 4.4.5 Power status register 1 (PWR_SR1)

Address offset: 0x10  
Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with the PWRRST bit in the APB peripheral reset register 1 (RCC_APBERRST1).

Access: 2 additional APB cycles are needed to read this register vs. a standard APB read.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WUF1</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>WUF6</td>
<td>WUF5</td>
<td>WUF4</td>
<td>r</td>
<td>r</td>
<td>WUF2</td>
<td>WUF1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **WUF1**: Wakeup flag internal  
This bit is set when a wakeup is detected on the internal wakeup line. It is cleared when all internal wakeup sources are cleared.

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **SBF**: Standby flag  
This bit is set by hardware when the device enters Standby mode and is cleared by setting the CSBF bit in the PWR_SCR register, or by a power-on reset. It is not cleared by the system reset.  
0: The device did not enter Standby mode  
1: The device entered Standby mode

Bits 7:6 Reserved, must be kept at reset value.
4.4.6 Power status register 2 (PWR_SR2)

Address offset: 0x14

Reset value: 0x0000 0000. This register is partially reset when exiting Standby/Shutdown modes.

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

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 PVDO: Power voltage detector output
- 0: VDD is above the selected PVD threshold
- 1: VDD is below the selected PVD threshold

Bit 10 VOSF: Voltage scaling flag
A delay is required for the internal regulator to be ready after the voltage scaling has been changed. VOSF indicates that the regulator reached the voltage level defined with VOS bits of the PWR_CR1 register.
- 0: The regulator is ready in the selected voltage range
- 1: The regulator output voltage is changing to the required voltage level
Bit 9 **REGLPF**: Low-power regulator flag

This bit is set by hardware when the MCU is in Low-power run mode. When the MCU exits the Low-power run mode, this bit remains at 1 until the regulator is ready in main mode. A polling on this bit must be done before increasing the product frequency.

This bit is cleared by hardware when the regulator is ready.

0: The regulator is ready in main mode (MR)
1: The regulator is in low-power mode (LPR)

Bit 8 **REGLPS**: Low-power regulator started

This bit provides the information whether the low-power regulator is ready after a power-on reset or Standby/Shutdown. If the Standby mode is entered while REGLPS bit is still cleared, the wakeup from Standby mode time may be increased.

0: The low-power regulator is not ready
1: The low-power regulator is ready

Bit 7 **FLASH_RDY**: Flash ready flag

This bit is set by hardware to indicate when the Flash memory is ready to be accessed after wakeup from power-down. To place the Flash memory in power-down, set either FPD_LPRUN, FPD_LPSLP or FPD_STP bits.

0: Flash memory in power-down
1: Flash memory ready to be accessed

**Note**: If the system boots from SRAM, the user application must wait till FLASH_RDY bit is set, prior to jumping to Flash memory.

Bits 6:0 Reserved, must be kept at reset value.

### 4.4.7 Power status clear register (PWR_SCR)

Address offset: 0x18

Reset value: 0x0000 0000.

Access: three additional APB cycles are needed to write this register, compared to a standard APB write.

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

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

<table>
<thead>
<tr>
<th>CSBF</th>
<th>CWUF5</th>
<th>CWUF4</th>
<th>CWUF3</th>
<th>CWUF2</th>
<th>CWUF1</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **CSBF**: Clear standby flag

Setting this bit clears the SBF flag in the PWR_SR1 register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **CWUF6**: Clear wakeup flag 6

Setting this bit clears the WUF6 flag in the PWR_SR1 register.

Bit 4 **CWUF5**: Clear wakeup flag 5

Setting this bit clears the WUF5 flag in the PWR_SR1 register.
### 4.4.8 Power Port A pull-up control register (PWR_PUCRA)

Address offset: 0x20.

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the APB peripheral reset register 1 (RCC_APB1RST1).

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>CWUF4</td>
<td>Clear wakeup flag 4. Setting this bit clears the WUF4 flag in the PWR_SR1 register.</td>
</tr>
<tr>
<td>2</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CWUF2</td>
<td>Clear wakeup flag 2. Setting this bit clears the WUF2 flag in the PWR_SR1 register.</td>
</tr>
<tr>
<td>0</td>
<td>CWUF1</td>
<td>Clear wakeup flag 1. Setting this bit clears the WUF1 flag in the PWR_SR1 register.</td>
</tr>
</tbody>
</table>

| Bit 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0  | PUy: Port A pull-up bit y (y=0..15) |

When set, this bit activates the pull-up on PA[y] when APC bit is set in PWR_CR3 register.

The pull-up is not activated if the corresponding PDy bit is also set.

### 4.4.9 Power Port A pull-down control register (PWR_PDCRA)

Address offset: 0x24.

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the APB peripheral reset register 1 (RCC_APB1RST1).

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

| Bit 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0  | PDy: Port A pull-down bit y (y=0..15) |

When set, this bit activates the pull-down on PA[y] when APC bit is set in PWR_CR3 register.

The pull-up is not activated if the corresponding PDy bit is also set.
4.4.10  Power Port B pull-up control register (PWR_PUCRB)

Address offset: 0x28.

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRST bit in the APB peripheral reset register 1 (RCC_APBRSTR1).

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PU15</td>
<td>PU14</td>
<td>PU13</td>
<td>PU12</td>
<td>PU11</td>
<td>PU10</td>
<td>PU9</td>
<td>PU8</td>
<td>PU7</td>
<td>PU6</td>
<td>PU5</td>
<td>PU4</td>
<td>PU3</td>
<td>PU2</td>
<td>PU1</td>
<td>PU0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  PDy: Port A pull-down bit y (y=0..15)

When set, this bit activates the pull-down on PA[y] when APC bit is set in PWR_CR3 register.

4.4.11  Power Port B pull-down control register (PWR_PDCRB)

Address offset: 0x2C.

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRST bit in the APB peripheral reset register 1 (RCC_APBRSTR1).

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PD15</td>
<td>PD14</td>
<td>PD13</td>
<td>PD12</td>
<td>PD11</td>
<td>PD10</td>
<td>PD9</td>
<td>PD8</td>
<td>PD7</td>
<td>PD6</td>
<td>PD5</td>
<td>PD4</td>
<td>PD3</td>
<td>PD2</td>
<td>PD1</td>
<td>PD0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  PDy: Port B pull-down bit y (y=0..15)

When set, this bit activates the pull-down on PB[y] when APC bit is set in PWR_CR3 register.
4.4.12  Power Port C pull-up control register (PWR_PUCRC)

Address offset: 0x30.

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the APB peripheral reset register 1 (RCC_APBRSTR1).

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PU15</td>
<td>PU14</td>
<td>PU13</td>
<td>PU12</td>
<td>PU11</td>
<td>PU10</td>
<td>PU9</td>
<td>PU8</td>
<td>PU7</td>
<td>PU6</td>
<td>PU5</td>
<td>PU4</td>
<td>PU3</td>
<td>PU2</td>
<td>PU1</td>
<td>PU0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0  

<table>
<thead>
<tr>
<th>PD15</th>
<th>PD14</th>
<th>PD13</th>
<th>PD12</th>
<th>PD11</th>
<th>PD10</th>
<th>PD9</th>
<th>PD8</th>
<th>PD7</th>
<th>PD6</th>
<th>PD5</th>
<th>PD4</th>
<th>PD3</th>
<th>PD2</th>
<th>PD1</th>
<th>PD0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0  

1. For STM32G031xx and STM32G041xx devices, y=6,7,13..15, as they do not have ports PC0 to PC5 and PC8 to PC12.

4.4.13  Power Port C pull-down control register (PWR_PDCRC)

Address offset: 0x34.

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the APB peripheral reset register 1 (RCC_APBRSTR1).

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PD15</td>
<td>PD14</td>
<td>PD13</td>
<td>PD12</td>
<td>PD11</td>
<td>PD10</td>
<td>PD9</td>
<td>PD8</td>
<td>PD7</td>
<td>PD6</td>
<td>PD5</td>
<td>PD4</td>
<td>PD3</td>
<td>PD2</td>
<td>PD1</td>
<td>PD0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0

1. For STM32G031xx and STM32G041xx devices, y=6,7,13..15, as they do not have ports PC0 to PC5 and PC8 to PC12.
4.4.14 Power Port D pull-up control register (PWR_PUCRD)

Address offset: 0x38.

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the APB peripheral reset register 1 (RCC_APBRSTR1).

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PU8</td>
<td>PU9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bits 9:8 **PUy**: Port D pull-up bit y (y=8, 9)(1)
When set, this bit activates the pull-up on PD[y] when APC bit is set in PWR_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

1. Reserved for STM32G031xx and STM32G041xx devices as they do not have ports PD8 and PD9.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **PUy**: Port D pull-up bit y (y=0..6)(1)
When set, this bit activates the pull-up on PD[y] when APC bit is set in PWR_CR3 register. The pull-up is not activated if the corresponding PDy bit is also set.

1. For STM32G031xx and STM32G041xx devices, y=0..3, as they do not have ports PD4 to PD6.

4.4.15 Power Port D pull-down control register (PWR_PDCRD)

Address offset: 0x3C.

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the APB peripheral reset register 1 (RCC_APBRSTR1).

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PD9</td>
<td>PD8</td>
<td>PD7</td>
<td>PD6</td>
<td>PD5</td>
<td>PD4</td>
<td>PD3</td>
<td>PD2</td>
<td>PD1</td>
<td>PD0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1) Reserved for STM32G031xx and STM32G041xx devices, y=0..3, as they do not have ports PD4 to PD6.
4.4.16 Power Port F pull-up control register (PWR_PUCRF)

Address offset: 0x48.

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the APB peripheral reset register 1 (RCC_APBRRST1).

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PU2</td>
<td>PU1</td>
<td>PU0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 2:0 PUy: Port F pull-up bit y (y=0..2)

When set, this bit activates the pull-up on PF[y] when APC bit is set in PWR_CR3 register.

The pull-up is not activated if the corresponding PDy bit is also set.

4.4.17 Power Port F pull-down control register (PWR_PDCRF)

Address offset: 0x4C.

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the APB peripheral reset register 1 (RCC_APBRRST1).

Access: Additional APB cycles are needed to access this register vs. a standard APB access (3 for a write and 2 for a read).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PD2</td>
<td>PD1</td>
<td>PD0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:3  Reserved, must be kept at reset value.

Bits 2:0  **P Dy**: Port F pull-down bit y (y=0..2)

When set, this bit activates the pull-down on PF[y] when APC bit is set in PWR_CR3 register.
## 4.4.18 PWR register map and reset value table

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>PWR_CR1</td>
<td>0x000</td>
<td>LPR, VOS, DBP, FPD_LPSLP, FPD_LPRUN, FPD_STOP, LPMS [2:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x004</td>
<td>VPDRT [2:0], VPDFT [2:0], VOS [1:0], DBP, FPD_LPSLP, FPD_LPRUN, FPD_STOP, LPMS [2:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x008</td>
<td>EMUL, APC, LPR, ULPEN, RRS, EMUL, EMUP, EMUP, EMUP, EMUP, EMUP, EMUP, EMUP, EMUP, EMUP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00C</td>
<td>RSE, VBE, WP0, WP1, WP2, WP3, WP4, WP5, WP6, WP7, WP8, WP9, WP10, WP11, WP12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x010</td>
<td>WUFI, SBF, WUF6, WUF5, WUF4, WUF3, WUF2, WUF1, WUF0, WUF9, WUF8, WUF7, WUF6, WUF5, WUF4, WUF3, WUF2, WUF1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x014</td>
<td>PVDRT [2:0], PVDFT [2:0], VOS [1:0], RST, CVF6, CVF7, CVF8, CVF9, CVF10, CVF11, CVF12, CVF13, CVF14, CVF15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x018</td>
<td>PVDRT [2:0], PVDFT [2:0], VOS [1:0], RST, CVF6, CVF7, CVF8, CVF9, CVF10, CVF11, CVF12, CVF13, CVF14, CVF15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x020</td>
<td>PVDO, VOSF, REGLP, REGLS, PVDRT [2:0], PVDFT [2:0], VOS [1:0], RST, CVF6, CVF7, CVF8, CVF9, CVF10, CVF11, CVF12, CVF13, CVF14, CVF15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x024</td>
<td>PVDRT [2:0], PVDFT [2:0], VOS [1:0], RST, CVF6, CVF7, CVF8, CVF9, CVF10, CVF11, CVF12, CVF13, CVF14, CVF15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x028</td>
<td>PVDRT [2:0], PVDFT [2:0], VOS [1:0], RST, CVF6, CVF7, CVF8, CVF9, CVF10, CVF11, CVF12, CVF13, CVF14, CVF15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x02C</td>
<td>PVDRT [2:0], PVDFT [2:0], VOS [1:0], RST, CVF6, CVF7, CVF8, CVF9, CVF10, CVF11, CVF12, CVF13, CVF14, CVF15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x030</td>
<td>PVDRT [2:0], PVDFT [2:0], VOS [1:0], RST, CVF6, CVF7, CVF8, CVF9, CVF10, CVF11, CVF12, CVF13, CVF14, CVF15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x034</td>
<td>PVDRT [2:0], PVDFT [2:0], VOS [1:0], RST, CVF6, CVF7, CVF8, CVF9, CVF10, CVF11, CVF12, CVF13, CVF14, CVF15</td>
</tr>
</tbody>
</table>
Table 29. PWR register map and reset values (continued)

| Offset | Register  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x038  | PWR_PUCRD |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x03C  | PWR_PDCRD |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x048  | PWR_PUCRF |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x04C  | PWR_PDCRF |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.2 on page 54 for the register boundary addresses.
5  Reset and clock control (RCC)

5.1  Reset

There are three types of reset, defined as system reset, power reset and RTC domain reset.

5.1.1  Power reset

A power reset is generated when one of the following events occurs:
- power-on reset (POR) or brown-out reset (BOR)
- exit from Standby mode
- exit from Shutdown mode

Power and brown-out reset set all registers to their reset values except the registers of the
RTC domain.

When exiting Standby mode, all registers in the V\textsubscript{CORE} domain are set to their reset value.
Registers outside the V\textsubscript{CORE} domain (RTC, WKUP, IWDG, and Standby/Shutdown mode
control) are not impacted.

When exiting Shutdown mode, the brown-out reset is generated, resetting all registers
except those in the RTC domain.

5.1.2  System reset

System reset sets all registers to their reset values except the reset flags in the clock
control/status register (RCC\_CSR) and the registers in the RTC domain.

System reset is generated when one of the following events occurs:
- low level on the NRST pin (external reset)
- window watchdog event (WWDG reset)
- independent watchdog event (IWDG reset)
- software reset (SW reset) (see Software reset)
- low-power mode security reset (see Low-power mode security reset)
- option byte loader reset (see Option byte loader reset)
- power-on reset

The reset source can be identified by checking the reset flags in the RCC\_CSR register
(see Section 5.4.22: Control/status register (RCC\_CSR)).

NRST pin (external reset):

Through specific option bits, the NRST pin is configurable for operating as:
- **Reset input/output** (default at device delivery)

Valid reset signal on the pin is propagated to the internal logic, and each internal reset
source is led to a pulse generator the output of which drives this pin. The GPIO
functionality (PF2) is not available. The pulse generator guarantees a minimum reset
pulse duration of 20 µs for each internal reset source to be output on the NRST pin. An
internal reset holder option can be used, if enabled in the option bytes, to ensure that
the pin is pulled low until its voltage meets V\text{IL} threshold. This function allows the
detection of internal reset sources by external components when the line faces a significant capacitive load.

- **Reset input**
  In this mode, any valid reset signal on the NRST pin is propagated to device internal logic, but resets generated internally by the device are not visible on the pin. In this configuration, GPIO functionality (PF2) is not available.

- **GPIO**
  In this mode, the pin can be used as PF2 standard GPIO. The reset function of the pin is not available. Reset is only possible from device internal reset sources and it is not propagated to the pin.

**Figure 9. Simplified diagram of the reset circuit**

Caution: Upon power reset or wakeup from shutdown mode, the NRST pin is configured as Reset input/output and driven low by the system until it is reconfigured to the expected mode when the option bytes are loaded, in the fourth clock cycle after the end of trstempo.

**Software reset**

The SYSRESETREQ bit in Cortex®-M0+ Application interrupt and reset control register must be set to force a software reset on the device (refer to the programming manual PM0223).
Low-power mode security reset
To prevent that critical applications mistakenly enter a low-power mode, three low-power
mode security resets are available. If enabled in option bytes, the resets are generated in
the following conditions:

- **Entering Standby mode**
  This type of reset is enabled by resetting nRST_STDBY bit in user option bytes. In this
  case, whenever a Standby mode entry sequence is successfully executed, the device
  is reset instead of entering Standby mode.

- **Entering Stop mode**
  This type of reset is enabled by resetting nRST_STOP bit in user option bytes. In this
  case, whenever a Stop mode entry sequence is successfully executed, the device is
  reset instead of entering Stop mode.

- **Entering Shutdown mode**
  This type of reset is enabled by resetting nRST_SHDW bit in user option bytes. In this
  case, whenever a Shutdown mode entry sequence is successfully executed, the device
  is reset instead of entering Shutdown mode.

For further information on the user option bytes, refer to Section 3.4.1: FLASH option byte
description.

Option byte loader reset
The option byte loader reset is generated when the OBL_LAUNCH bit (bit 27) is set in the
FLASH_CR register. This bit is used to launch the option byte loading by software.

5.1.3 RTC domain reset
The RTC domain has two specific resets.
A RTC domain reset is generated when one of the following events occurs:

- **Software reset**, triggered by setting the BDRST bit in the RTC domain control register
  *(RCC_BDCR)*.
- **V_DD or V_BAT power on**, if both supplies have previously been powered off.

A RTC domain reset only affects the LSE oscillator, the RTC, the backup registers and the
RCC RTC domain control register.

5.2 Clocks
The device provides the following clock sources producing primary clocks:

- **HSI RC** - a high-speed fully-integrated RC oscillator producing HSI16 clock (about
  16 MHz)
- **HSE OSC** - a high-speed oscillator with external crystal/ceramic resonator or external
  clock source, producing HSE clock (4 to 48 LHz)
- **LSI RC** - a low-speed fully-integrated RC oscillator producing LSI clock (about 32 kHz)
- **LSE OSC** - a low-speed oscillator with external crystal/ceramic resonator or external
  clock source, producing LSE clock (accurate 32.768 kHz or external clock up to 1 MHz)
- **I2S_CKIN** - pin for direct clock input for I2S1 peripheral
Each oscillator can be switched on or off independently when it is not used, to optimize power consumption. Check sub-sections of this section for more functional details. For electrical characteristics of the internal and external clock sources, refer to the device datasheet.

The device produces secondary clocks by dividing or/multiplying the primary clocks:

- **HSISYS** - a clock derived from HSI16 through division by a factor programmable from 1 to 128
- **PLLPCLK**, **PLLPCLK** and **PLLRCLK** - clocks output from the PLL block
- **SYSCLK** - a clock obtained through selecting one of LSE, LSI, HSE, PLLRCLK, and HSISYS clocks
- **HCLK** - a clock derived from SYSCLK through division by a factor programmable from 1 to 512
- **HCLK8** - a clock derived from HCLK through division by eight
- **PCLK** - a clock derived from HCLK through division by a factor programmable from 1 to 16
- **TIMPCLK** - a clock derived from PCLK, running at PCLK frequency if the APB prescaler division factor is set to 1, or at twice the PCLK frequency otherwise
- **LPTIMx_IN** - clock from LPTIMx_INx pins, selectable for the LPTIM peripheral

More secondary clocks are generated by fixed division of HSE, HSI16 and HCLK clocks.

The HSISYS is used as system clock source after startup from reset, with the division by 1 (producing HSI16 frequency).

The HCLK clock and PCLK clock are used for clocking the AHB and the APB domains, respectively. Their maximum allowed frequency is 64 MHz.

The peripherals are clocked with the clocks from the bus they are attached to (HCLK for AHB, PCLK for APB) except:

- **TIMx**, with these clock sources to select from:
  - TIMPCLK (selectable for all timers) running at PCLK frequency if the APB prescaler division factor is set to 1, or at twice the PCLK frequency otherwise
  - PLLQCLK selectable for high-speed TIM1 and TIM15 timers

- **LPTIMx**, with these clock sources to select from:
  - LSI
  - LSE
  - HSI16
  - PCLK (APB clock)
  - LPTIMx_IN selected from LPTIMx_INx pins

The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE.

- **UCPD**, always clocked with HSI16
- **ADC**, with these clock sources to select from:
  - SYSCLK (system clock)
  - HSI16
  - PLLPCLK
- **USARTx / LPUART1**, with these clock sources to select from:
  - SYSCLK (system clock)
  - HSI16
  - LSE
  - PCLK (APB clock)
The wakeup from Stop mode is supported only when the clock is HSI16 or LSE.

- **I2Cx**, with these clock sources to select from:
  - SYSCLK (system clock)
  - HSI16
  - PCLK (APB clock)
The wakeup from Stop mode is supported only when the clock is HSI16.

- **I2S1**, with these clock sources to select from:
  - SYSCLK (system clock)
  - HSI16
  - PLLPCLK
  - I2S_CKIN pin

- **RNG**, with these clock sources to select from:
  - SYSCLK (system clock)
  - HSI16 clock divided by 8
  - PLLQCLK
The RNG clock can additionally be divided by 2, 4 or 8, using a dedicated prescaler.

- **CEC**, with these clock sources to select from:
  - HSI16 clock divided by 488
  - LSE

- **RTC**, with these clock sources to select from:
  - LSE
  - LSI
  - HSE clock divided by 32
The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE.

- **IWDG**, always clocked with LSI clock.

- **SysTick** (Cortex® core system timer), with these clock sources to select from:
  - HCLK (AHB clock)
  - HCLK clock divided by 8
The selection is done through SysTick control and status register.

HCLK is used as Cortex®-M0+ free-running clock (FCLK). For more details, refer to the programming manual PM0223.
1. Only applies to STM32G071xx and STM32G081xx.
5.2.1 HSE clock

The high speed external clock signal (HSE) can be generated from two possible clock sources:

- HSE external crystal/ceramic resonator
- HSE user external clock

The resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

![Figure 11. HSE/ LSE clock sources](image)

<table>
<thead>
<tr>
<th>Clock source</th>
<th>Hardware configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>External clock</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>Crystal/Ceramic resonators</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>
External crystal/ceramic resonator (HSE crystal)

The 4 to 48 MHz external oscillator has the advantage of producing a very accurate rate on the main clock.

The associated hardware configuration is shown in Figure 11. Refer to the electrical characteristics section of the datasheet for more details.

The HSERDY flag in the Clock control register (RCC_CR) indicates if the HSE oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the Clock interrupt enable register (RCC_CIER).

The HSE Crystal can be switched on and off using the HSEON bit in the Clock control register (RCC_CR).

External source (HSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 48 MHz. This mode is selected by setting the HSEBYP and HSEON bits in the Clock control register (RCC_CR). The external clock signal (square, sinus or triangle) with ~40-60 % duty cycle depending on the frequency (refer to the datasheet) must drive the OSC_IN pin, on devices where OSC_IN and OSC_OUT pins are available (see Figure 11). The OSC_OUT pin can be used as a GPIO.

The OSC_OUT pin can be used as a GPIO or it can be configured as OSC_EN alternate function, to provide an enable signal to external clock synthesizer. It allows stopping the external clock source when the device enters low power modes.

Note: For details on pin availability, refer to the pinout section in the corresponding device datasheet.

To minimize the consumption, it is recommended to use the square signal.

5.2.2 HSI16 clock

The HSI16 clock signal is generated from an internal 16 MHz RC oscillator.

The HSI16 RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator. However, even after calibration, it is less accurate than an oscillator using a frequency reference such as quartz crystal or ceramic resonator.

The HSISYS clock derived from HSI16 can be selected as system clock after wakeup from Stop modes (Stop 0 or Stop 1). Refer to Section 5.3: Low-power modes. It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to Section 5.2.8: Clock security system (CSS).

Calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. To compensate for this variation, each device is factory calibrated to 1 % accuracy at $T_A=25^\circ C$.

After reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the Internal clock sources calibration register (RCC_ICSCR).

Voltage or temperature variations in the application may affect the HSI16 frequency of the RC oscillator. It can be trimmed using the HSITRIM[6:0] bits in the Internal clock sources calibration register (RCC_ICSCR).
For more details on how to measure the HSI16 frequency variation, refer to Section 5.2.15: Internal/external clock measurement with TIM14/TIM16/TIM17.

The HSIRDY flag in the Clock control register (RCC_CR) indicates if the HSI16 RC is stable or not. At startup, the HSI16 RC output clock is not released until this bit is set by hardware.

The HSI16 RC can be switched on and off using the HSION bit in the Clock control register (RCC_CR).

The HSI16 signal can also be used as a backup source (auxiliary clock) if the HSE crystal oscillator fails. Refer to Section 5.2.8: Clock security system (CSS) on page 152.

### 5.2.3 PLL

The internal PLL multiplies the frequency of HSI16- or HSE-based clock fetched on its input, to produce three independent clock outputs. The allowed input frequency range is from 2.66 to 16 MHz. The dedicated divider PLLM with division factor programmable from one to eight allows setting a frequency within the valid PLL input range. Refer to Figure 10: Clock tree and PLL configuration register (RCC_PLLCFGR).

The PLL configuration (selection of the input clock and multiplication factor) must be done before enabling the PLL. Once the PLL is enabled, these parameters cannot be changed.

To modify the PLL configuration, proceed as follows:

1. Disable the PLL by setting PLLON to 0 in Clock control register (RCC_CR).
2. Wait until PLLRDY is cleared. The PLL is now fully stopped.
3. Change the desired parameter.
4. Enable the PLL again by setting PLLON to 1.
5. Enable the desired PLL outputs by configuring PLLPEN, PLLQEN, and PLLREN in PLL configuration register (RCC_PLLCFGR).

An interrupt can be generated when the PLL is ready, if enabled in the Clock interrupt enable register (RCC_CIER).

The enable bit of each PLL output clock (PLLPEN, PLLQEN, and PLLREN) can be modified at any time without stopping the PLL. PLLREN cannot be cleared if PLLRCLK is used as system clock.

### 5.2.4 LSE clock

The LSE crystal is a 32.768 kHz crystal or ceramic resonator. It has the advantage of providing a low-power but highly accurate clock source to the real-time clock peripheral (RTC) for clock/calendar or other timing functions.

The LSE crystal is switched on and off using the LSEON bit in RTC domain control register (RCC_BDCR). The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits in the RTC domain control register (RCC_BDCR) to obtain the best compromise between robustness and short start-up time on one side and low-power-consumption on the other side. The LSE drive can be decreased to the lower drive capability (LSEDRV=00) when the LSE is ON. However, once LSEDRV is selected, the drive capability can not be increased if LSEON=1.

The LSERDY flag in the RTC domain control register (RCC_BDCR) indicates whether the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the Clock interrupt enable register (RCC_CIER).
External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. This mode is selected by setting the LSEBYP and LSEON bits in the AHB peripheral clock enable in Sleep/Stop mode register (RCC_AHBSMENR). The external clock signal (square, sinus or triangle) with ~50 % duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin can be used as GPIO. See Figure 11.

5.2.5 LSI clock

The LSI RC acts as a low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG) and RTC. The clock frequency is 32 kHz. For more details, refer to the electrical characteristics section of the datasheets.

The LSI RC can be switched on and off using the LSION bit in the Control/status register (RCC_CSR).

The LSIRDY flag in the Control/status register (RCC_CSR) indicates if the LSI oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the Clock interrupt enable register (RCC_CIER).

5.2.6 System clock (SYSCLK) selection

One of the following clocks can be selected as system clock (SYSCLK):

- LSI
- LSE
- HSISYS
- HSE
- PLLRCLK

The system clock maximum frequency is 64 MHz. Upon system reset, the HSISYS clock derived from HSI16 oscillator is selected as system clock. When a clock source is used directly or through the PLL as a system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source which is not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in the Internal clock sources calibration register (RCC_ICSCR) indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

5.2.7 Clock source frequency versus voltage scaling

The following table gives the different clock source frequencies depending on the product voltage range.

<table>
<thead>
<tr>
<th>Clock</th>
<th>Maximum clock frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Range 1</td>
</tr>
<tr>
<td>HSI16</td>
<td>16</td>
</tr>
<tr>
<td>HSE</td>
<td>48</td>
</tr>
<tr>
<td>PLLPCLK</td>
<td>122(1)</td>
</tr>
</tbody>
</table>
5.2.8 Clock security system (CSS)

Clock security system can be activated by software. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped. If a failure is detected on the HSE clock:

- the HSE oscillator is automatically disabled
- a clock failure event is sent to the break input of TIM1, TIM15, TIM16 and TIM17 timers
- CSSI (clock security system interrupt) is generated

The CSSI is linked to the Cortex®-M0+ NMI (non-maskable interrupt) exception vector. It makes the software aware of a HSE clock failure to allow it to perform rescue operations.

Note: If the CSS is enabled and the HSE clock fails, the CSSI occurs and an NMI is automatically generated. The NMI is executed infinitely unless the CSS interrupt pending bit is cleared. It is therefore necessary that the NMI ISR clears the CSSI by setting the CSSC bit in the Clock interrupt clear register (RCC_CICR).

If HSE is selected directly or indirectly (PLLRCLK selected for SYSCLK and HSE selected as PLL input) as system clock, and a failure of HSE clock is detected, the system clock switches automatically to HSISYS and the HSE oscillator is disabled. If the HSE clock (divided or not) is the clock entry of the PLL and PLLRCLK is used as system clock when the failure occurs, the PLL is disabled, too.

5.2.9 Clock security system for LSE clock (LSECSS)

A clock security system on LSE can be activated by setting the LSECSSON bit in RTC domain control register (RCC_BDCR). This bit can be cleared only by a hardware reset or RTC software reset, or after LSE clock failure detection. LSECSSON must be written after LSE and LSI are enabled (LSEON and LSION enabled) and ready (LSERDY and LSIRDY flags set by hardware), and after selecting the RTC clock by RTCSEL.

The LSECSS works in all modes except VBAT. It keeps working also under system reset (excluding power-on reset). If a failure is detected on the LSE oscillator, the LSE clock is no longer supplied to the RTC but its registers are not impacted.

Note: If the LSECSS is enabled and the LSE clock fails, the LSECSSI occurs and an NMI is automatically generated. The NMI is executed infinitely unless the LSECSS interrupt pending bit is cleared. It is therefore necessary that the NMI ISR clears the LSECSSI by setting the LSECSSC bit in the Clock interrupt clear register (RCC_CICR).

If LSE is used as system clock, and a failure of LSE clock is detected, the system clock switches automatically to LSI. In low-power modes, an LSE clock failure generates a wakeup. The interrupt flag must then be cleared within the RCC registers.

### Table 30. Clock source frequency (continued)

<table>
<thead>
<tr>
<th>Clock</th>
<th>Maximum clock frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Range 1</td>
</tr>
<tr>
<td>PLLQCLK</td>
<td>128(1)</td>
</tr>
<tr>
<td>PLLRCLK</td>
<td>64(1)</td>
</tr>
</tbody>
</table>

1. Maximum VCO frequency is 344 MHz.
2. Maximum VCO frequency is 128 MHz.
The software **must** then disable the LSECSSON bit, stop the defective 32 kHz oscillator (by clearing LSEON), and change the RTC clock source (no clock, LSI or HSE, with RTCSEL), or take any appropriate action to secure the application.

The frequency of the LSE oscillator must exceed 30 kHz to avoid false positive detections.

### 5.2.10 ADC clock

The ADC clock is derived from the system clock, or from the PLLPCLK output. It can reach 122 MHz and can be divided by the following prescalers values: 1,2,4,6,8,10,12,16,32,64,128 or 256 by configuring the ADC1_CCR register. It is asynchronous to the AHB clock. Alternatively, the ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). This programmable factor is configured using the CKMODE bit fields in the ADC1_CCR.

If the programmed factor is 1, the AHB prescaler must be set to 1.

### 5.2.11 RTC clock

The RTCCLK clock source can be either the HSE/32, LSE or LSI clock. It is selected by programming the RTCSEL[1:0] bits in the **RTC domain control register (RCC_BDCR)**. This selection cannot be modified without resetting the RTC domain. The system must always be configured so as to get a PCLK frequency greater then or equal to the RTCCLK frequency for a proper operation of the RTC.

The LSE clock is in the RTC domain, whereas the HSE and LSI clocks are not. Consequently:

- If LSE is selected as RTC clock:
  - The RTC continues to work even if the V\textsubscript{DD} supply is switched off, provided the V\textsubscript{BAT} supply is maintained.
- If LSI is selected as the RTC clock:
  - The RTC state is not guaranteed if the V\textsubscript{DD} supply is powered off.
- If the HSE clock divided by a prescaler is used as the RTC clock:
  - The RTC state is not guaranteed if the V\textsubscript{DD} supply is powered off or if the internal voltage regulator is powered off (removing power from the V\textsubscript{CORE} domain).

When the RTC clock is LSE or LSI, the RTC remains clocked and functional under system reset.

### 5.2.12 Timer clock

The timer clock TIMPCLK is derived from PCLK (used for APB) as follows:

1. If the APB prescaler is set to 1, TIMPCLK frequency is equal to PCLK frequency.
2. Otherwise, the TIMPCLK frequency is set to twice the PCLK frequency.

For TIM1 and TIM15, PLLQCLK clock can also be selected, if:

- PCLK is derived from PLLRCLK, and
- PLLQCLK frequency is an integer multiplication by 2 or more of the PCLK frequency, without exceeding 128 MHz.
5.2.13 Watchdog clock

If the Independent watchdog (IWDG) is started by either hardware option or software access, the LSI oscillator is forced ON and cannot be disabled. After the LSI oscillator temporization, the clock is provided to the IWDG.

5.2.14 Clock-out capability

MCO

The microcontroller clock output (MCO) capability allows the clock to be output onto the external MCO pin. One of the following can be selected as the MCO clock:
- LSI
- LSE
- SYSCLK
- HSI16
- HSE
- PLLRCLK

The selection is controlled by the MCOSEL[3:0] bits of the Clock configuration register (RCC_CFGR). The selected clock can be divided with a factor programmable through the MCOPRE[2:0] field of Clock configuration register (RCC_CFGR).

LSCO

The LSCO pin allows outputting one of low-speed clocks:
- LSI
- LSE

The selection is controlled by the LSCOSEL, and enabled with the LSCOEN in the RTC domain control register (RCC_BDCR). The configuration registers of the corresponding GPIO port must be programmed in alternate function mode.

This function remains available in Stop 0, Stop 1 and Standby modes.

5.2.15 Internal/external clock measurement with TIM14/TIM16/TIM17

It is possible to indirectly measure the frequency of all on-board clock sources with the TIM14, TIM16 and TIM17 channel 1 input capture, as represented in Figure 12, Figure 13 and Figure 14.

TIM14

By setting the TI1SEL[3:0] field of the TIM14_TISEL register, the clock selected for the input capture channel 1 of TIM14 can be one of:
- GPIO (refer to the alternate function mapping in the device datasheets)
- RTC clock (RTCCCLK)
- HSE clock divided by 32
- MCO (MCU clock output)

The last option is controlled by the MCOSEL[3:0] field of the clock configuration register (RCC_CFGR). All clock sources can be selected for the MCO pin.
TIM16
By setting the TI1SEL[3:0] field of the TIM16_TISEL register, the clock selected for the input capture channel1 of TIM16 can be one of:
- GPIO (refer to the alternate function mapping in the device datasheets).
- LSI clock
- LSE clock
- RTC wakeup interrupt signal
The last option requires to enable the RTC interrupt.

TIM17
By setting the TI1SEL[3:0] field of the TIM17_TISEL register, the clock selected for the input capture channel1 of TIM17 can be one of:
- GPIO Refer to the alternate function mapping in the device datasheets.
- HSE divided by 32
- MCO (MCU clock output)
The last option is controlled by the MCOSEL[3:0] field of the clock configuration register (RCC_CFGR). All clock sources can be selected for the MCO pin.
Calibration of the HSI16 oscillator

For TIM14, TIM15 and TIM17, the primary purpose of connecting the LSE to the channel 1 input capture is to be able to precisely measure the HSISYS clock (derived from HSI16) selected as system clock. Counting HSISYS clock pulses between consecutive edges of the LSE clock (the time reference) allows measuring the HSISYS (and HSI16) clock period. Such measurement can determine the HSI16 oscillator frequency with nearly the same accuracy as the accuracy of the 32.768 kHz quartz crystal used with the LSE oscillator (typically a few tens of ppm). The HSI16 oscillator can then be trimmed to compensate for deviations from target frequency, due to manufacturing, process, temperature and/or voltage variation.

The HSI16 oscillator has dedicated user-accessible calibration bits for this purpose.

The basic concept consists in providing a relative measurement (for example, the HSISYS/LSE ratio): the measurement accuracy is therefore closely related to the ratio between the two clock sources. Increasing the ratio allows improving the measurement accuracy.

Generated by the HSE oscillator, the HSE clock (divided by 32) used as time reference is the second best method for reaching a good HSI16 frequency measurement accuracy. It is recommended in absence of the LSE clock.

In order to further improve the precision of the HSI16 oscillator calibration, it is advised to employ one or a combination of the following measures to increase the frequency measurement accuracy:

- set the HSISYS divider to 1 for HSISYS frequency to be equal to HSI16 frequency
- average the results of multiple consecutive measurements
- use the input capture prescaler of the timer (one capture every up to eight periods)
- use LSE clock for the RTC and the RTC wakeup interrupt signal as time reference

The last point significantly increases the reference period for HSI16 clock pulse counting, which improves the accuracy of a single measurement. For operation, the RTC wakeup interrupt must be enabled.

Calibration of the LSI oscillator

The calibration of the LSI oscillator uses the same principle as that for calibrating the HSI16 oscillator. TIM16 channel1 input capture must be used for LSI clock, and HSE selected as system clock source. The number of HSE clock pulses between consecutive edges of the LSI signal, counted by TIM16, is then representative of the LSI clock period.
5.2.16 Peripheral clock enable registers

Each peripheral clock can be enabled by the corresponding enable bit of the RCC_AHBENR or RCC_APBENRx registers.

When the peripheral clock is not active, the peripheral registers read or write accesses are not supported.

Caution: The enable bit has a synchronization mechanism to create a glitch-free clock for the peripheral. After the enable bit is set, there is a 2-clock-cycle delay before the clock be active, which the software must take into account.

5.3 Low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep and Low Power Sleep modes stops the CPU clock. The memory interface clocks (Flash memory and SRAM interfaces) can be stopped by software during sleep mode. The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.
- Stop modes (Stop 0 and Stop 1) stop all the clocks in the V\text{CORE} domain and disables the PLL as well as the HSI16 and HSE oscillators.
  
The USART1, USART2, LPUART1, and I2C1 peripherals can enable the HSI16 oscillator even when the MCU is in Stop mode (if HSI16 is selected as clock source for that peripheral).
  
The LPUART1, USART1, and USART2 peripherals can also operate with the clock from the LSE oscillator when the system is in Stop mode, if LSE is selected as clock source for that peripheral and the LSE oscillator is enabled (LSEON set). In that case, the LSE oscillator remains active when the device enters Stop mode (these peripherals do not have the capability to turn on the LSE oscillator).
- Standby and Shutdown modes stop all clocks in the V\text{CORE} domain and disable the PLL, as well as the HSI16 and HSE oscillators.
  
The CPU’s deepsleep mode can be overridden for debugging by setting the DBG_STOP or DBG\_STANDBY bits in the DBGMCU\_CR register.
  
When leaving the Stop 0 or Stop 1 modes, HSISYS becomes automatically the system clock.

When leaving the Standby and Shutdown modes, HSISYS (with frequency equal to HSI16) becomes automatically the system clock. At wakeup from Standby and Shutdown mode, the user trim is lost.

If a Flash memory programming operation is ongoing, Stop, Standby, and Shutdown entry is delayed until the Flash memory interface access is finished. If an access to the APB domain is ongoing, the Stop, Standby, and Shutdown entry is delayed until the APB access is finished.
5.4  RCC registers

5.4.1  Clock control register (RCC_CR)

Address offset: 0x00

Power-on reset value: 0x0000 0500

Other types of reset: same as power-on reset, except HSEBYP bit that keeps its previous value

Access: no wait state, word, half-word and byte access

<table>
<thead>
<tr>
<th>Bit</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rw</td>
<td>PLLRDY: PLL clock ready flag. Set by hardware to indicate that the PLL is locked. 0: PLL unlocked 1: PLL locked</td>
</tr>
<tr>
<td>30</td>
<td>rw</td>
<td>PLLON: PLL enable. Set and cleared by software to enable the PLL. Cleared by hardware when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the PLL clock is used as the system clock. 0: PLL OFF 1: PLL ON</td>
</tr>
<tr>
<td>29</td>
<td>rw</td>
<td>HSIDIV[2:0]: HSI clock divider.</td>
</tr>
<tr>
<td>28</td>
<td>rw</td>
<td>HSI RDY: HSI clock ready flag. Set by hardware when the HSE oscillator is ready, and disabled by hardware if a HSE clock failure is detected. This bit is set only and is cleared by reset. 0: HSI OFF 1: HSI ON</td>
</tr>
<tr>
<td>27</td>
<td>rw</td>
<td>HSI KERON: Clock security system enable. Set by software to enable the clock security system. When CSSON is set, the clock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if a HSE clock failure is detected. This bit is set only and is cleared by reset. 0: Clock security system OFF (clock detector OFF) 1: Clock security system ON (Clock detector ON if the HSE oscillator is stable, OFF if not).</td>
</tr>
<tr>
<td>26</td>
<td>rw</td>
<td>HSEBYP: HSE crystal oscillator bypass. Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit set, to be used by the device. The HSEBYP bit can be written only if the HSE oscillator is disabled. 0: HSE crystal oscillator not bypassed 1: HSE crystal oscillator bypassed with external clock</td>
</tr>
</tbody>
</table>

Bits 31:26  Reserved, must be kept at reset value.

Bits 23:20  Reserved, must be kept at reset value.
Reset and clock control (RCC)

Bit 17 **HSERDY**: HSE clock ready flag
Set by hardware to indicate that the HSE oscillator is stable.
0: HSE oscillator not ready
1: HSE oscillator ready

*Note: Once the HSEON bit is cleared, HSERDY goes low after 6 HSE clock cycles.*

Bit 16 **HSEON**: HSE clock enable
Set and cleared by software.
Cleared by hardware to stop the HSE oscillator when entering Stop, Standby, or Shutdown mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.
0: HSE oscillator OFF
1: HSE oscillator ON

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:11 **HSIDIV[2:0]**: HSI16 clock division factor
This bit field controlled by software sets division factor of the HSI16 clock divider to produce HSISYS clock:
- 000: 1
- 001: 2
- 010: 4
- 011: 8
- 100: 16
- 101: 32
- 110: 64
- 111: 128

Bit 10 **HSIRDY**: HSI16 clock ready flag
Set by hardware to indicate that HSI16 oscillator is stable. This bit is set only when HSI16 is enabled by software by setting HSION.
0: HSI16 oscillator not ready
1: HSI16 oscillator ready

*Note: Once the HSION bit is cleared, HSIRDY goes low after 6 HSI16 clock cycles.*

Bit 9 **HSIKERON**: HSI16 always enable for peripheral kernels.
Set and cleared by software to force HSI16 ON even in Stop modes. The HSI16 can only feed USART1, USART2, CEC and I2C1 peripherals configured with HSI16 as kernel clock. Keeping the HSI16 ON in Stop mode allows avoiding to slow down the communication speed because of the HSI16 startup time. This bit has no effect on HSION value.
0: No effect on HSI16 oscillator.
1: HSI16 oscillator is forced ON even in Stop mode.

Bit 8 **HSION**: HSI16 clock enable
Set and cleared by software.
Cleared by hardware to stop the HSI16 oscillator when entering Stop, Standby, or Shutdown mode.
Forced by hardware to keep the HSI16 oscillator ON when it is used directly or indirectly as system clock (also when leaving Stop, Standby, or Shutdown modes, or in case of failure of the HSE oscillator used for system clock).
0: HSI16 oscillator OFF
1: HSI16 oscillator ON

Bits 7:0 Reserved, must be kept at reset value.
5.4.2 Internal clock sources calibration register (RCC_ICSCR)

Address offset: 0x04
Reset value: 0x0000 40XX, where X is factory-programmed.
Access: no wait state, word, half-word and byte access

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

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **HSITRIM[6:0]**: HSI16 clock trimming
These bits provide an additional user-programmable trimming value that is added to the HSICAL[7:0] bits. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the HSI16 clock.
The default value is 64, which, when added to the HSICAL value, should trim the HSI16 to 16 MHz ± 1 %.

Bits 7:0 **HSICAL[7:0]**: HSI16 clock calibration
These bits are initialized at startup with the factory-programmed HSI16 calibration trim value. When HSITRIM is written, HSICAL is updated with the sum of HSITRIM and the factory trim value.

5.4.3 Clock configuration register (RCC_CFG)

Address offset: 0x08
Reset value: 0x0000 0000
Access: 0 ≤ wait state ≤ 2, word, half-word and byte access
1 or 2 wait states inserted only if the access occurs during clock source switch.
From 0 to 15 wait states inserted if the access occurs when the APB or AHB prescalers values update is on going.

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

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWS[2:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**MCOPRE[2:0]**
**MCOSEL[2:0]**
**PPRE[2:0]**
**HPRE[3:0]**
Bit 31  Reserved, must be kept at reset value.

Bits 30:28  **MCOPRE[2:0]:** Microcontroller clock output prescaler  
This bit field is controlled by software. It sets the division factor of the clock sent to the MCO output as follows:
000: 1  
001: 2  
010: 4  
011: 8  
100: 16  
101: 32  
110: 64  
111: 128  
It is highly recommended to set this field before the MCO output is enabled.

Bit 27  Reserved, must be kept at reset value.

Bits 26:24  **MCOSEL[2:0]:** Microcontroller clock output  
This bit field is controlled by software. It sets the clock selector for MCO output as follows:
000: no clock, MCO output disabled  
001: SYSCLK  
010: Reserved  
011: HSI16  
100: HSE  
101: PLLRCLK  
110: LSI  
111: LSE  
*Note:* This clock output may have some truncated cycles at startup or during MCO clock source switching.

Bits 23:15  Reserved, must be kept at reset value.

Bits 14:12  **PPRE[2:0]:** APB prescaler  
This bit field is controlled by software. To produce PCLK clock, it sets the division factor of HCLK clock as follows:
0xx: 1  
100: 2  
101: 4  
110: 8  
111: 16
5.4.4 PLL configuration register (RCC_PLLCFGR)

Address offset: 0x0C
Reset value: 0x0000 1000
Access: no wait state, word, half-word and byte access

This register is used to configure the PLL clock outputs according to the formulas:

- \( f_{VCO} = f_{PLLIN} \times \frac{N}{M} \)
- \( f_{PLL} = f_{VCO} / P \)
- \( f_{PLLQ} = f_{VCO} / Q \)
• \( f_{PLL} = \frac{f_{VCO}}{R} \)

where \( f_{PLLIN} \) is the PLL input clock frequency, \( f_{VCO} \) is the PLL VCO frequency, and \( P \), \( Q \) and \( R \) are \( f_{VCO} \) division factors and \( f_{PLL} \), \( f_{PLLQ} \) and \( f_{PLLR} \) the clock frequencies of the PLLPCLK, PLLQCLK and PLLRCLK PLL clock outputs, respectively.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:29</td>
<td>PLLR[2:0]: PLL VCO division factor R for PLLRCLK clock output</td>
</tr>
<tr>
<td>28</td>
<td>PLLREN: PLLRCLK clock output enable</td>
</tr>
<tr>
<td>27:25</td>
<td>PLLQ[2:0]: PLL VCO division factor Q for PLLQCLK clock output</td>
</tr>
<tr>
<td>24</td>
<td>PLLQEN: PLLQCLK clock output enable</td>
</tr>
</tbody>
</table>

The bit field can be written only when the PLL is disabled.

The PLLRCLK clock can be selected as system clock.

**Caution:** The software must set this bit field so as not to exceed 64 MHz on this clock.

This bit is controlled by software to enable/disable the PLLRCLK clock output of the PLL:

- 0: Disable
- 1: Enable

This bit cannot be written when PLLRCLK output of the PLL is selected for system clock. Disabling the PLLRCLK clock output, when not used, allows saving power.

The bit field can be written only when the PLL is disabled.

**Caution:** The software must set this bitfield so as not to exceed 128 MHz on this clock.

This bit is controlled by software to enable/disable the PLLQCLK clock output of the PLL:

- 0: Disable
- 1: Enable

Disabling the PLLQCLK clock output, when not used, allows saving power.
Bits 23:22 Reserved, must be kept at reset value.

Bits 21:17 **PLLP[4:0]**: PLL VCO division factor P for PLLPCLK clock output
   This bit field is controlled by software. It sets the PLL VCO division factor P as follows:
   00000: Reserved
   00001: 2
   ...
   11111: 32
   The bit field can be written only when the PLL is disabled.
   **Caution**: The software must set this bit field so as not to exceed 122 MHz on this clock.

Bit 16 **PLLPEN**: PLLPCLK clock output enable
   This bit is controlled by software to enable/disable the PLLPCLK clock output of the PLL:
   0: Disable
   1: Enable
   Disabling the PLLPCLK clock output, when not used, allows saving power.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PLLN[6:0]**: PLL frequency multiplication factor N
   This bit is controlled by software to set the division factor of the \( f_{VCO} \) feedback divider (that
determines the PLL multiplication ratio) as follows:
   0000000: Invalid
   0000001: Reserved
   ...
   0000111: Reserved
   0001000: 8
   0001001: 9
   ...
   1010101: 85
   1010110: 86
   1010111: Reserved
   ...
   1111111: Reserved
   The bit field can be written only when the PLL is disabled.
   **Caution**: The software must set these bits so that the VCO output frequency is between 64
   and 344 MHz.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PLLM**: Division factor M of the PLL input clock divider
   This bit is controlled by software to divide the PLL input clock before the actual phase-locked
   loop, as follows:
   00: 1
   01: 2
   10: 3
   11: 4
   100: 5
   101: 6
   110: 7
   111: 8
   The bit field can be written only when the PLL is disabled.
   **Caution**: The software must set these bits so that the PLL input frequency after the /M divider
   is between 4 and 16 MHz.
Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 PLLSRC: PLL input clock source

This bit is controlled by software to select PLL clock source, as follows:

00: No clock
01: Reserved
10: HSI16
11: HSE

The bit field can be written only when the PLL is disabled.

When the PLL is not used, selecting 00 allows saving power.

5.4.5 Clock interrupt enable register (RCC_CIER)

Address offset: 0x18
Reset value: 0x0000 0000
Access: no wait state, word, half-word and byte access

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

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

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

Bit 5 PLLRDYIE: PLL ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by PLL lock:

0: Disable
1: Enable

Bit 4 HSERDYIE: HSE ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSE oscillator stabilization:

0: Disable
1: Enable

Bit 3 HSIIRDYIE: HSI16 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSI16 oscillator stabilization:

0: Disable
1: Enable
5.4.6 Clock interrupt flag register (RCC_CIFR)

Address offset: 0x1C
Reset value: 0x0000 0000
Access: no wait state, word, half-word and byte access

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

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

Bit 9 LSECSSF: LSE clock security system interrupt flag
Set by hardware when a failure is detected in the LSE oscillator.
Cleared by software setting the LSECSSC bit.
0: No clock security interrupt caused by LSE clock failure
1: Clock security interrupt caused by LSE clock failure

Bit 8 CSSF: HSE clock security system interrupt flag
Set by hardware when a failure is detected in the HSE oscillator.
Cleared by software setting the CSSC bit.
0: No clock security interrupt caused by HSE clock failure
1: Clock security interrupt caused by HSE clock failure

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 PLLRDYF: PLL ready interrupt flag
Set by hardware when the PLL locks and PLLRDYDIE is set.
Cleared by software setting the PLLRDYCYC bit.
0: No clock ready interrupt caused by PLL lock
1: Clock ready interrupt caused by PLL lock
Bit 4 **HSERDYF**: HSE ready interrupt flag
- Set by hardware when the HSE clock becomes stable and HSERDYDIE is set.
- Cleared by software setting the HSERDYC bit.
- 0: No clock ready interrupt caused by the HSE oscillator
- 1: Clock ready interrupt caused by the HSE oscillator

Bit 3 **HSIRDYF**: HSI16 ready interrupt flag
- Set by hardware when the HSI16 clock becomes stable and HSIRDYDIE is set in a response to setting the HSI16 oscillator through a clock request, this bit is not set and no interrupt is generated.
- Cleared by software setting the HSIRDYC bit.
- 0: No clock ready interrupt caused by the HSI16 oscillator
- 1: Clock ready interrupt caused by the HSI16 oscillator

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYF**: LSE ready interrupt flag
- Set by hardware when the LSE clock becomes stable and LSERDYDIE is set.
- Cleared by software setting the LSERDYC bit.
- 0: No clock ready interrupt caused by the LSE oscillator
- 1: Clock ready interrupt caused by the LSE oscillator

Bit 0 **LSIRDYF**: LSI ready interrupt flag
- Set by hardware when the LSI clock becomes stable and LSIRDYDIE is set.
- Cleared by software setting the LSIRDYC bit.
- 0: No clock ready interrupt caused by the LSI oscillator
- 1: Clock ready interrupt caused by the LSI oscillator

### 5.4.7 Clock interrupt clear register (RCC_CICR)

Address offset: 0x20
Reset value: 0x0000 0000
Access: no wait state, word, half-word and byte access

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

Address: 0x24
Reset value: 0x0000 0000
Access: no wait state, word, half-word and byte access
5.4.9 **AHB peripheral reset register (RCC_AHBRSTR)**

Address offset: 0x28

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

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

1. Available on STM32G08xxx devices only.

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **RNGRST**: Random number generator reset

Set and cleared by software.

0: No effect

1: Reset RNG

Bits 17 Reserved, must be kept at reset value.
Bit 16 **AESRST**: AES hardware accelerator reset
Set and cleared by software.
0: No effect
1: Reset AES

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCRST**: CRC reset
Set and cleared by software.
0: No effect
1: Reset CRC

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHRST**: Flash memory interface reset
Set and cleared by software.
0: No effect
1: Reset Flash memory interface
This bit can only be set when the Flash memory is in power down mode.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **DMARST**: DMA and DMAMUX reset
Set and cleared by software.
0: No effect
1: Reset DMA and DMAMUX

### 5.4.10 APB peripheral reset register 1 (RCC_APBRSTR1)

Address offset: 0x2C
Reset value: 0x0000 0000
Access: no wait state, word, half-word and byte access

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM1 RST</td>
<td>LPTIM2 RST</td>
<td>DAC1 RST</td>
<td>PWR RST</td>
<td>DBG RST</td>
<td>UCPD2 RST</td>
<td>UCPD1 RST</td>
<td>CEC RST</td>
<td>Res. I2C2 RST</td>
<td>I2C1 RST</td>
<td>LP UART1 RST</td>
<td>USART4 RST</td>
<td>USART3 RST</td>
<td>USART2 RST</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:12</th>
<th>Bits 11:8</th>
<th>Bits 7:4</th>
<th>Bits 3:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

170/1234 RM0444 Rev 2
Bit 31  **LPTIM1RST**: Low Power Timer 1 reset  
Set and cleared by software.  
0: No effect  
1: Reset LPTIM1

Bit 30  **LPTIM2RST**: Low Power Timer 2 reset  
Set and cleared by software.  
0: No effect  
1: Reset LPTIM2

Bit 29  **DAC1RST**: DAC1 interface reset.  
Set and cleared by software.  
0: No effect  
1: Reset DAC1 interface  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 28  **PWRRST**: Power interface reset  
Set and cleared by software.  
0: No effect  
1: Reset PWR

Bit 27  **DBGRST**: Debug support reset  
Set and cleared by software.  
0: No effect  
1: Reset DBG

Bit 26  **UCPD2RST**: UCPD2 reset  
Set and cleared by software.  
0: No effect  
1: Reset UCPD2  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 25  **UCPD1RST**: UCPD1 reset  
Set and cleared by software.  
0: No effect  
1: Reset UCPD1  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 24  **CECRST**: HDMI CEC reset  
Set and cleared by software.  
0: No effect  
1: Reset the HDMI CEC  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 23  Reserved, must be kept at reset value.

Bit 22  **I2C2RST**: I2C2 reset  
Set and cleared by software.  
0: No effect  
1: Reset I2C2
Bit 21  **I2C1RST**: I2C1 reset
Set and cleared by software.
0: No effect
1: Reset I2C1

Bit 20  **LPUART1RST**: LPUART1 reset
Set and cleared by software.
0: No effect
1: Reset LPUART1

Bit 19  **USART4RST**: USART4 reset
Set and cleared by software.
0: No effect
1: Reset USART4
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 18  **USART3RST**: USART3 reset
Set and cleared by software.
0: No effect
1: Reset USART3
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 17  **USART2RST**: USART2 reset
Set and cleared by software.
0: No effect
1: Reset USART2

Bits 16:15  Reserved, must be kept at reset value.

Bit 14  **SPI2RST**: SPI2 reset
Set and cleared by software.
0: No effect
1: Reset SPI2

Bits 13:6  Reserved, must be kept at reset value.

Bit 5  **TIM7RST**: TIM7 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM7
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 4  **TIM6RST**: TIM6 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM6
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.
5.4.11 APB peripheral reset register 2 (RCC_APBRSTR2)

Address offset: 0x30
Reset value: 0x0000 0000
Access: no wait state, word, half-word and byte access

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

Bit 15 **TIM14RST**: TIM14 timer reset  
0: No effect  
1: Reset TIM14 timer

Bit 14 **USART1RST**: USART1 reset  
0: No effect  
1: Reset USART1

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1RST**: SPI1 reset  
0: No effect  
1: Reset SPI1

Bit 11 **TIM1RST**: TIM1 timer reset  
0: No effect  
1: Reset TIM1 timer

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGRST**: SYSCFG, COMP and VREFBUF reset  
0: No effect  
1: Reset SYSCFG + COMP + VREFBUF

## 5.4.12 I/O port clock enable register (RCC_IOPENR)

Address: 0x34  
Reset value: 0x0000 0000  
Access: no wait state, word, half-word and byte access

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

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

Bit 5 **GPIOFEN**: I/O port F clock enable  
This bit is set and cleared by software.  
0: Disable  
1: Enable
Bit 4 Reserved, must be kept at reset value.

Bit 3 **GPIODEN**: I/O port D clock enable
This bit is set and cleared by software.
0: Disable
1: Enable

Bit 2 **GPIOCEN**: I/O port C clock enable
This bit is set and cleared by software.
0: Disable
1: Enable

Bit 1 **GPIOBEN**: I/O port B clock enable
This bit is set and cleared by software.
0: Disable
1: Enable

Bit 0 **GPIOAEN**: I/O port A clock enable
This bit is set and cleared by software.
0: Disable
1: Enable

### 5.4.13 AHB peripheral clock enable register (RCC_AHBENR)

**Address offset**: 0x38

**Reset value**: 0x00000 0100

**Access**: no wait state, word, half-word and byte access

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

1. Available on STM32G08xxx devices only.

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **RNGEN**: Random number generator clock enable
Set and cleared by software.
0: Disable
1: Enable

Bits 17 Reserved, must be kept at reset value.

Bit 16 **AESEN**: AES hardware accelerator
Set and cleared by software.
0: Disable
1: Enable

Bits 15:13 Reserved, must be kept at reset value.
Bit 12 **CRCEN**: CRC clock enable  
Set and cleared by software.  
0: Disable  
1: Enable  

Bits 11:9 Reserved, must be kept at reset value.  

Bit 8 **FLASHEN**: Flash memory interface clock enable  
Set and cleared by software.  
0: Disable  
1: Enable  
This bit can only be cleared when the Flash memory is in power down mode.  

Bits 7:1 Reserved, must be kept at reset value.  

Bit 0 **DMAEN**: DMA and DMAMUX clock enable  
Set and cleared by software.  
0: Disable  
1: Enable

### 5.4.14 APB peripheral clock enable register 1 (RCC_APBENR1)

Address offset: 0x3C  
Reset value: 0x0000 0000  
Access: no wait state, word, half-word and byte access

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>LPTIM1EN: LPTIM1 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>LPTIM2EN: LPTIM2 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>DAC1EN: DAC1 interface clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>PWR EN: PWR clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>DBG EN: DBG clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>UCPD2EN: UCPD2 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>UCPD1EN: UCPD1 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>CEC EN: CEC clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>I2C2 EN: I2C2 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>I2C1 EN: I2C1 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>LP UART1 EN: LP UART1 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>USART4 EN: USART4 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>USART3 EN: USART3 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>USART2 EN: USART2 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>SPI2EN: SPI2 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>WWDG EN: WWDG clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>RTC EN: RTC clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>APB EN: APB clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>TIM7 EN: TIM7 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TIM6 EN: TIM6 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TIM3 EN: TIM3 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>TIM2 EN: TIM2 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **LPTIM1EN**: LPTIM1 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable  

Bit 30 **LPTIM2EN**: LPTIM2 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable  

Bit 29 **DAC1EN**: DAC1 interface clock enable  
Set and cleared by software.  
0: Disable  
1: Enable  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.
Bit 28 **PWREN**: Power interface clock enable  
Set and cleared by software.  
0: Disable  
1: Enable

Bit 27 **DBGEN**: Debug support clock enable  
Set and cleared by software.  
0: Disable  
1: Enable

Bit 26 **UCPD2EN**: UCPD2 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 25 **UCPD1EN**: UCPD1 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 24 **CECEN**: HDMI CEC clock enable  
Set and cleared by software.  
0: Disable  
1: Enable  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **I2C2EN**: I2C2 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable

Bit 21 **I2C1EN**: I2C1 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable

Bit 20 **LPUART1EN**: LPUART1 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable

Bit 19 **USART4EN**: USART4 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.
Bit 18 **USART3EN**: USART3 clock enable
Set and cleared by software.
- 0: Disable
- 1: Enable
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 17 **USART2EN**: USART2 clock enable
Set and cleared by software.
- 0: Disable
- 1: Enable

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **SPI2EN**: SPI2 clock enable
Set and cleared by software.
- 0: Disable
- 1: Enable

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGEN**: WWDG clock enable
Set by software to enable the window watchdog clock. Cleared by hardware system reset
- 0: Disable
- 1: Enable
This bit can also be set by hardware if the WWDG_SW option bit is 0.

Bit 10 **RTCAPBEN**: RTC APB clock enable
Set and cleared by software.
- 0: Disable
- 1: Enable

Bits 9:6 Reserved, must be kept at reset value.

Bit 5 **TIM7EN**: TIM7 timer clock enable
Set and cleared by software.
- 0: Disable
- 1: Enable
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 4 **TIM6EN**: TIM6 timer clock enable
Set and cleared by software.
- 0: Disable
- 1: Enable
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.
Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM3EN**: TIM3 timer clock enable
Set and cleared by software.
0: Disable
1: Enable

Bit 0 **TIM2EN**: TIM2 timer clock enable
Set and cleared by software.
0: Disable
1: Enable

5.4.15 **APB peripheral clock enable register 2 (RCC_APBENR2)**

Address offset: 0x40
Reset value: 0x0000 0000
Access: no wait state, word, half-word and byte access

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

| rw    | rw    | rw   | rw    | rw    | rw   | rw   |

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **ADCEN**: ADC clock enable
Set and cleared by software.
0: Disable
1: Enable

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM17EN**: TIM16 timer clock enable
Set and cleared by software.
0: Disable
1: Enable

Bit 17 **TIM16EN**: TIM16 timer clock enable
Set and cleared by software.
0: Disable
1: Enable

Bit 16 **TIM15EN**: TIM15 timer clock enable
Set and cleared by software.
0: Disable
1: Enable
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.
Bit 15 **TIM14EN**: TIM14 timer clock enable  
Set and cleared by software.  
0: Disable  
1: Enable  

Bit 14 **USART1EN**: USART1 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN**: SPI1 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable

Bit 11 **TIM1EN**: TIM1 timer clock enable  
Set and cleared by software.  
0: Disable  
1: Enable

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGEN**: SYSCFG, COMP and VREFBUF clock enable  
Set and cleared by software.  
0: Disable  
1: Enable

5.4.16 **I/O port in Sleep mode clock enable register (RCC_IOPSMENR)**

Address: 0x44  
Reset value: 0x0000 002F

Access: no wait state, word, half-word and byte access

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

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

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

Bit 5 **GPIOFSMEN**: I/O port F clock enable during Sleep mode  
Set and cleared by software.  
0: Disable  
1: Enable
Bit 4  Reserved, must be kept at reset value.

Bit 3  GPIODSMEN: I/O port D clock enable during Sleep mode
       Set and cleared by software.
       0: Disable
       1: Enable

Bit 2  GPIOCSMEN: I/O port C clock enable during Sleep mode
       Set and cleared by software.
       0: Disable
       1: Enable

Bit 1  GPIOBSMEN: I/O port B clock enable during Sleep mode
       Set and cleared by software.
       0: Disable
       1: Enable

Bit 0  GPIOASMEN: I/O port A clock enable during Sleep mode
       Set and cleared by software.
       0: Disable
       1: Enable

### 5.4.17 AHB peripheral clock enable in Sleep/Stop mode register (RCC_AHBSMENR)

Address offset: 0x48

Reset value: 0x0005 1301

Access: no wait state, word, half-word and byte access

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

1. Available on STM32G08xxx devices only.

Bits 31:19  Reserved, must be kept at reset value.

Bit 18  RNGSMEN: Random number generator clock enable during Sleep and Stop mode
       Set and cleared by software.
       0: Disable
       1: Enable

Bits 17  Reserved, must be kept at reset value.

Bit 16  AESSMEN: AES hardware accelerator clock enable during Sleep mode
       Set and cleared by software.
       0: Disable
       1: Enable
5.4.18 **APB peripheral clock enable in Sleep/Stop mode register 1 (RCC_APBSMENR1)**

Address offset: 0x4C

Reset value: 0xFF7E 4C33

Access: no wait state, word, half-word and byte access

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM1 SMEN</td>
<td>LPTIM2 SMEN</td>
<td>DAC1 SMEN</td>
<td>PWR SMEN</td>
<td>DBG SMEN</td>
<td>UCPD2 SMEN</td>
<td>UCPD1 SMEN</td>
<td>CEC SMEN</td>
<td>Res.</td>
<td>I2C2 SMEN</td>
<td>I2C1 SMEN</td>
<td>LP UART1 SMEN</td>
<td>USART4 SMEN</td>
<td>USART3 SMEN</td>
<td>USART2 SMEN</td>
<td>Res.</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:13: Reserved, must be kept at reset value.

Bit 12 **CRCSMEN**: CRC clock enable during Sleep mode

Set and cleared by software.

0: Disable
1: Enable

Bits 11:10: Reserved, must be kept at reset value.

Bit 9 **SRAMSMEN**: SRAM clock enable during Sleep mode

Set and cleared by software.

0: Disable
1: Enable

Bit 8 **FLASHSMEN**: Flash memory interface clock enable during Sleep mode

Set and cleared by software.

0: Disable
1: Enable

This bit can be activated only when the Flash memory is in power down mode.

Bits 7:1: Reserved, must be kept at reset value.

Bit 0 **DMASMEN**: DMA and DMAMUX clock enable during Sleep mode

Set and cleared by software.

0: Disable
1: Enable
Bit 31 **LPTIM1SMEN**: Low Power Timer 1 clock enable during Sleep and Stop modes
   Set and cleared by software.
   0: Disable
   1: Enable

Bit 30 **LPTIM2SMEN**: Low Power Timer 2 clock enable during Sleep and Stop modes
   Set and cleared by software.
   0: Disable
   1: Enable

Bit 29 **DAC1SMEN**: DAC1 interface clock enable during Sleep and Stop modes
   Set and cleared by software.
   0: Disable
   1: Enable
   This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 28 **PWRSMEN**: Power interface clock enable during Sleep mode
   Set and cleared by software.
   0: Disable
   1: Enable

Bit 27 **DBGSMEN**: Debug support clock enable during Sleep mode
   Set and cleared by software.
   0: Disable
   1: Enable

Bit 26 **UCPD2SMEN**: UCPD2 clock enable during Sleep mode
   Set and cleared by software.
   0: Disable
   1: Enable
   This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 25 **UCPD1SMEN**: UCPD1 clock enable during Sleep mode
   Set and cleared by software.
   0: Disable
   1: Enable
   This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 24 **CECSMEN**: HDMI CEC clock enable during Sleep and Stop modes
   Set and cleared by software.
   0: Disable
   1: Enable
   This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **I2C2SMEN**: I2C2 clock enable during Sleep mode
   Set and cleared by software.
   0: Disable
   1: Enable
Bit 21 **I2C1SMEN**: I2C1 clock enable during Sleep and Stop modes  
Set and cleared by software.  
0: Disable  
1: Enable  

Bit 20 **LPUART1SMEN**: LPUART1 clock enable during Sleep and Stop modes  
Set and cleared by software.  
0: Disable  
1: Enable  

Bit 19 **USART4SMEN**: USART4 clock enable during Sleep mode  
Set and cleared by software.  
0: Disable  
1: Enable  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.  

Bit 18 **USART3SMEN**: USART3 clock enable during Sleep mode  
Set and cleared by software.  
0: Disable  
1: Enable  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.  

Bit 17 **USART2SMEN**: USART2 clock enable during Sleep and Stop modes  
Set and cleared by software.  
0: Disable  
1: Enable  

Bits 16:15 Reserved, must be kept at reset value.  

Bit 14 **SPI2SMEN**: SPI2 clock enable during Sleep mode  
Set and cleared by software.  
0: Disable  
1: Enable  

Bits 13:12 Reserved, must be kept at reset value.  

Bit 11 **WWDGSMEN**: WWDG clock enable during Sleep and Stop modes  
Set and cleared by software.  
0: Disable  
1: Enable  

Bit 10 **RTCAPBSMEN**: RTC APB clock enable during Sleep mode  
Set and cleared by software.  
0: Disable  
1: Enable  

Bits 9:6 Reserved, must be kept at reset value.  

Bit 5 **TIM7SMEN**: TIM7 timer clock enable during Sleep mode  
Set and cleared by software.  
0: Disable  
1: Enable  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.
Bit 4 **TIM6SMEN**: TIM6 timer clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM3SMEN**: TIM3 timer clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable

Bit 0 **TIM2SMEN**: TIM2 timer clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable

5.4.19 **APB peripheral clock enable in Sleep/Stop mode register 2 (RCC_APB2SMENR2)**

Address offset: 0x50
Reset value: 0x0017D801
Access: no wait state, word, half-word and byte access

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

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **ADCSMEN**: ADC clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM17SMEN**: TIM16 timer clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable

Bit 17 **TIM16SMEN**: TIM16 timer clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
Bit 16 **TIM15SMEN**: TIM15 timer clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 15 **TIM14SMEN**: TIM14 timer clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable

Bit 14 **USART1SMEN**: USART1 clock enable during Sleep and Stop modes
Set and cleared by software.
0: Disable
1: Enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1SMEN**: SPI1 clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable

Bit 11 **TIM1SMEN**: TIM1 timer clock enable during Sleep mode
Set and cleared by software.
0: Disable
1: Enable

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGSMEN**: SYSCFG, COMP and VREFBUF clock enable during Sleep and Stop modes
Set and cleared by software.
0: Disable
1: Enable

### 5.4.20 Peripherals independent clock configuration register (RCC_CCIPR)

**Address**: 0x54

**Reset value**: 0x0000 0000

**Access**: no wait states, word, half-word and byte access

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>ADCSEL[1:0] RNGDIV[1:0] RNGSEL[1:0] Res. TIM15 SEL Res. TIM1 SEL LPTIM2SEL[1:0] LPTIM1SEL[1:0] Res. Res.</td>
<td>rw rw rw rw rw rw rw rw rw rw rw</td>
<td>0x0000 0000</td>
</tr>
</tbody>
</table>

**Example**: To enable the TIM15 timer clock during Sleep mode, set bit 16 to 1. To enable the ADCSEL[1:0] register, set bits 31:30 to 0x02.
Bits 31:30  **ADCSEL[1:0]**: ADCs clock source selection  
This bit field is controlled by software to select the clock source for ADC:  
00: System clock  
01: PLLPCLK  
10: HSI16  
11: Reserved  

Bits 29:28  **RNGDIV[1:0]**: Division factor of RNG clock divider  
This bit field is controlled by software to select the division factor as follows:  
00: 1  
01: 2  
10: 4  
11: 8  

Bits 27:26  **RNGSEL[1:0]**: RNG clock source selection  
This bit field is controlled by software to select the RNG clock as follows:  
00: No clock  
01: HSI16  
10: SYSCLK  
11: PLLQCLK  

Bit 25  Reserved, must be kept at reset value.  

Bit 24  **TIM15SEL**: TIM15 clock source selection  
This bit is set and cleared by software. It selects TIM15 clock source as follows:  
0: TIMPCLK  
1: PLLQCLK  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.  

Bit 23  Reserved, must be kept at reset value.  

Bit 22  **TIM1SEL**: TIM1 clock source selection  
This bit is set and cleared by software. It selects TIM1 clock source as follows:  
0: TIMPCLK  
1: PLLQCLK  

Bits 21:20  **LPTIM2SEL[1:0]**: LPTIM2 clock source selection  
This bit field is controlled by software to select LPTIM2 clock source as follows:  
00: PCLK  
01: LSI  
10: HSI16  
11: LSE  

Bits 19:18  **LPTIM1SEL[1:0]**: LPTIM1 clock source selection  
This bit field is controlled by software to select LPTIM1 clock source as follows:  
00: PCLK  
01: LSI  
10: HSI16  
11: LSE  

Bits 17:16  Reserved, must be kept at reset value.
5.4.21 RTC domain control register (RCC_BDCR)

Address offset: 0x5C
Reset value: 0x0000 0000, reset by RTC domain reset, except LSCOSEL, LSCOEN and BDRST which are reset only by RTC domain power-on reset.
Access: 0 ≤ wait state ≤ 3, word, half-word and byte access
Wait states are inserted in case of successive accesses to this register.
Note: The bits of the RTC domain control register (RCC_BDCR) are outside of the V\textsubscript{CORE} domain. As a result, after Reset, these bits are write-protected and the DBP bit in the Section 4.4.1: Power control register 1 (PWR_CR1) has to be set before these can be modified. Refer to Section 4.1.2: Battery backup of RTC domain on page 103 for further information. These bits (except LSCOSEL, LSCOEN and BDRST) are only reset after a RTC domain Reset (see Section 5.1.3: RTC domain reset). Any internal or external Reset will not have any effect on these bits.

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

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

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **LSCOSEL**: Low-speed clock output selection
Set and cleared by software to select the low-speed output clock:
0: LSI
1: LSE

Bit 24 **LSCOEN**: Low-speed clock output (LSCO) enable
Set and cleared by software.
0: Disable
1: Enable

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **BDRST**: RTC domain software reset
Set and cleared by software to reset the RTC domain:
0: No effect
1: Reset

Bit 15 **RTCEN**: RTC clock enable
Set and cleared by software. The bit enables clock to RTC and TAM:
0: Disable
1: Enable

Bits 14:10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]**: RTC clock source selection
Set by software to select the clock source for the RTC as follows:
00: No clock
01: LSE
10: LSI
11: HSE divided by 32

Once the RTC clock source is selected, it cannot be changed anymore unless the RTC domain is reset, or unless a failure is detected on LSE (LSECSSD is set). The BDRST bit can be used to reset this bit field to 00.

Bit 7 Reserved, must be kept at reset value.
5.4.22 Control/status register (RCC_CSR)

Address: 0x60

Reset value: 0xXX00 0000, reset by system Reset, except reset flags by power Reset only.
Access: 0 ≤ wait state ≤ 3, word, half-word and byte access
Wait states are inserted in case of successive accesses to this register.

Bit 6  **LSECSSD** CSS on LSE failure Detection
Set by hardware to indicate when a failure is detected by the clock security system on the external 32 kHz oscillator (LSE):
0: No failure detected
1: Failure detected

Bit 5  **LSECSSON** CSS on LSE enable
Set by software to enable the clock security system on LSE (32 kHz) oscillator as follows:
0: Disable
1: Enable
LSECSSON must be enabled after the LSE oscillator is enabled (LSEON bit enabled) and ready (LSERDY flag set by hardware), and after the RTCSEL bit is selected.
Once enabled, this bit cannot be disabled, except after a LSE failure detection (LSECSSD = 1). In that case the software must disable the LSECSSON bit.

Bits 4:3  **LSEDRV[1:0]** LSE oscillator drive capability
Set by software to select the LSE oscillator drive capability as follows:
00: low driving capability
01: medium-low driving capability
10: medium-high driving capability
11: high driving capability
Applicable when the LSE oscillator is in Xtal mode, as opposed to bypass mode.

Bit 2  **LSEBYP**: LSE oscillator bypass
Set and cleared by software to bypass the LSE oscillator (in debug mode).
0: Not bypassed
1: Bypassed
This bit can be written only when the external 32 kHz oscillator is disabled (LSEON=0 and LSERDY=0).

Bit 1  **LSERDY**: LSE oscillator ready
Set and cleared by hardware to indicate when the external 32 kHz oscillator is ready (stable):
0: Not ready
1: Ready
After the LSEON bit is cleared, LSERDY goes low after 6 external low-speed oscillator clock cycles.

Bit 0  **LSEON**: LSE oscillator enable
Set and cleared by software to enable LSE oscillator:
0: Disable
1: Enable
### Bit 31 **LPWRSTF**: Low-power reset flag
Set by hardware when a reset occurs due to illegal Stop, Standby, or Shutdown mode entry.
Cleared by setting the RMVF bit.
- 0: No illegal mode reset occurred
- 1: Illegal mode reset occurred

This operates only if nRST_STOP, nRST_STDBY or nRST_SHDW option bits are cleared.

### Bit 30 **WWDGRSTF**: Window watchdog reset flag
Set by hardware when a window watchdog reset occurs.
Cleared by setting the RMVF bit.
- 0: No window watchdog reset occurred
- 1: Window watchdog reset occurred

### Bit 29 **IWDGRSTF**: Independent window watchdog reset flag
Set by hardware when an independent watchdog reset domain occurs.
Cleared by setting the RMVF bit.
- 0: No independent watchdog reset occurred
- 1: Independent watchdog reset occurred

### Bit 28 **SFTRSTF**: Software reset flag
Set by hardware when a software reset occurs.
Cleared by setting the RMVF bit.
- 0: No software reset occurred
- 1: Software reset occurred

### Bit 27 **PWRRSTF**: BOR or POR/PDR flag
Set by hardware when a BOR or POR/PDR occurs.
Cleared by setting the RMVF bit.
- 0: No BOR or POR occurred
- 1: BOR or POR occurred

### Bit 26 **PINRSTF**: Pin reset flag
Set by hardware when a reset from the NRST pin occurs.
Cleared by setting the RMVF bit.
- 0: No reset from NRST pin occurred
- 1: Reset from NRST pin occurred

### Bit 25 **OBLRSTF**: Option byte loader reset flag
Set by hardware when a reset from the Option byte loading occurs.
Cleared by setting the RMVF bit.
- 0: No reset from Option byte loading occurred
- 1: Reset from Option byte loading occurred

### Bit 24 Reserved, must be kept at reset value.
5.4.23 RCC register map

The following table gives the RCC register map and the reset values.

Table 31. RCC register map and reset values

| Offset | Register   | Bits 31 | Bits 30 | Bits 29 | Bits 28 | Bits 27 | Bits 26 | Bits 25 | Bits 24 | Bits 23 | Bits 22 | Bits 21 | Bits 20 | Bits 19 | Bits 18 | Bits 17 | Bits 16 | Bits 15 | Bits 14 | Bits 13 | Bits 12 | Bits 11 | Bits 10 | Bits 9  | Bits 8  | Bits 7  | Bits 6  | Bits 5  | Bits 4  | Bits 3  | Bits 2  | Bits 1  | Bits 0  |
|--------|------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
|       | Reset value| 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     | 0 0     |
|       | Reset value| 1 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0x04   |
|       | Reset value| 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0x08   |
|       | Reset value| 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0x0C   |
| 0x10   | Reserved   | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0x10   |
| 0x14   | Reserved   | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0x14   |
Table 31. RCC register map and reset values (continued)

| Offset | Register      | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x18   | RCC_CIER      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x1C   | RCC_CIFR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x20   | RCC_CICR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x24   | RCC_IOPRSTR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x28   | RCC_AHBRSTR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2C   | RCC_APBRSTR1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x30   | RCC_APBRSTR2  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x34   | RCC_IOPENR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Off-set: Address offset from the RCC register base
Register: Name of the register
Reset value: Initial value of the register on reset
Table 31. RCC register map and reset values (continued)

| Offset | Register          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x38   | RCC_AHBENR        | Res| Res| Res| Res| Res| Res| Res| Res| Res| Res| Res| Res| Res| Res| Res| Res| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |   |   |
Refer to Section 2.2 on page 54 for the register boundary addresses.
6 General-purpose I/Os (GPIO)

6.1 Introduction
Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR) and a 32-bit set/reset register (GPIOx_BSRR). In addition all GPIOs have a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

6.2 GPIO main features
- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers (at most 8 AFs possible per I/O)
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

6.3 GPIO functional description
Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:
- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIOx_BSRR and GPIOx_BRR registers is to allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.
Figure 15 shows the basic structures of a standard I/O port bit. Table 32 gives the possible port bit configurations.

**Figure 15. Basic structure of an I/O port bit**

**Table 32. Port bit configuration table (1)**

<table>
<thead>
<tr>
<th>MODE(i) [1:0]</th>
<th>OTYPE(i)</th>
<th>OSPEED(i) [1:0]</th>
<th>PUPD(i) [1:0]</th>
<th>I/O configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td></td>
<td>SPEED [1:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>GP output PP</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>GP output PP + PU</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>GP output PP + PD</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>GP output OD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>GP output OD + PU</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>GP output OD + PD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Reserved (GP output OD)</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>SPEED [1:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>AF</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>AF</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>AF</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>AF</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>AF</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>AF</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
6.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in analog mode.

The debug pins are in AF pull-up/pull-down after reset:
- PA14: SWCLK in pull-down
- PA13: SWDIO in pull-up

Note: PA14 is shared with BOOT0 functionality. Caution is required as the debugging device can manipulate BOOT0 pin value.

Upon reset, the UCPD CCx lines present a pull-down resistor that can be disabled by setting the UCPDx_STROBE bit of the SYSCFG_CFGR1 register.

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIOx_PUPDR register.

6.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals available on the same I/O pin.

Table 32. Port bit configuration table(1) (continued)

<table>
<thead>
<tr>
<th>MODE(1) [1:0]</th>
<th>OTYPE(i)</th>
<th>OSPEED(i) [1:0]</th>
<th>PUPD(i) [1:0]</th>
<th>I/O configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>x</td>
<td>x x</td>
<td>0 0</td>
<td>Input Floating</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x x</td>
<td>0 1</td>
<td>Input PU</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x x</td>
<td>1 0</td>
<td>Input PD</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x x</td>
<td>1 1</td>
<td>Reserved (input floating)</td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x x</td>
<td>0 0</td>
<td>Input/output Analog</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x x</td>
<td>0 1</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x x</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x x</td>
<td>1 1</td>
<td></td>
</tr>
</tbody>
</table>

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.
Each I/O pin has a multiplexer with up to eight alternate function inputs (AF0 to AF7) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user has to proceed as follows:

- **Debug function**: after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host
- **GPIO**: configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- **Peripheral alternate function**:  
  - Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDER registers, respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Additional functions**:  
  - ADC, DAC and COMP connection can be enabled in ADC, DAC or COMP registers regardless the configured GPIO mode. When ADC, DAC or COMP uses a GPIO, it is recommended to configure the GPIO in analog mode, through the GPIOx_MODER register.
  - For the additional functions like RTC, TAMP, WKUPx and oscillators, configure the required function in the related RTC, TAMP, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

### 6.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

### 6.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR). GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.
6.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register which allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, bit BS(i) sets the corresponding ODR(i) bit. When written to 1, bit BR(i) resets the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: it is possible to modify one or more bits in a single atomic AHB write access.

6.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write / read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence has been applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence (refer to Section 6.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to D, F)) can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details refer to LCKR register description in Section 6.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to D, F).

6.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can
thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

6.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the given pin must not be configured in analog mode or being used as oscillator pin, so the input trigger is kept enabled.

Refer to Section 12: Extended interrupt and event controller (EXTI).

6.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register provides the I/O state

Figure 16 shows the input configuration of the I/O port bit.

**Figure 16. Input floating/pull up/pull down configurations**
6.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state
- A read access to the output data register gets the last written value

Figure 17 shows the output configuration of the I/O port bit.

![Output configuration diagram](image)

6.3.11 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data)
- The Schmitt trigger input is activated
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state

Figure 18 shows the Alternate function configuration of the I/O port bit.
6.3.12 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up and pull-down resistors are disabled by hardware
- Read access to the input data register gets the value “0”

*Figure 19* shows the high-impedance, analog-input configuration of the I/O port bits.
6.3.13 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched OFF (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched ON (by setting the HSEON or LSEON bit in the RCC_CSR register) the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the OSC_IN or OSC32_IN pin is reserved for clock input and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

6.3.14 Using the GPIO pins in the RTC domain

The PC13/PC14/PC15 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to .

6.3.15 USB PD / Dead battery support

The content of this section will be provided later.
6.4 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to Table 33.

The peripheral registers can be written in word, half word or byte mode.

6.4.1 GPIO port mode register (GPIOx_MODER) (x = A to D, F)

Address offset: 0x00

Reset value:
- 0xEBFF FFFF for port A
- 0xFFFF FFFF for other ports

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

Bits 31:0 **MODE[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
- 00: Input mode
- 01: General purpose output mode
- 10: Alternate function mode
- 11: Analog mode (reset state)

6.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to D, F)

Address offset: 0x04

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>OT15</td>
<td>OT14</td>
<td>OT13</td>
<td>OT12</td>
<td>OT11</td>
<td>OT10</td>
<td>OT9</td>
<td>OT8</td>
<td>OT7</td>
<td>OT6</td>
<td>OT5</td>
<td>OT4</td>
<td>OT3</td>
<td>OT2</td>
<td>OT1</td>
<td>OT0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OT[15:0]**: Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O output type.
- 0: Output push-pull (reset state)
- 1: Output open-drain
6.4.3  **GPIO port output speed register (GPIOx_OSPEEDR)**  
(x = A to D, F)  

Address offset: 0x08  
Reset value: 0xC00 0000 (for port A)  
Reset value: 0x0000 0000 (for other ports)  

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSPEED15</td>
<td>OSPEED14</td>
<td>OSPEED13</td>
<td>OSPEED12</td>
<td>OSPEED11</td>
<td>OSPEED10</td>
<td>OSPEED9</td>
<td>OSPEED8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **OSPEED[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)  
These bits are written by software to configure the I/O output speed.  
00: Very low speed  
01: Low speed  
10: High speed  
11: Very high speed  

*Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.*  
The FT_c GPIOs cannot be set to high speed.

6.4.4  **GPIO port pull-up/pull-down register (GPIOx_PUPDR)**  
(x = A to D, F)  

Address offset: 0x0C  
Reset value: 0x2400 0000 (for port A)  
Reset value: 0x0000 0000 (for other ports)  

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **PUPD[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)  
These bits are written by software to configure the I/O pull-up or pull-down  
00: No pull-up, pull-down  
01: Pull-up  
10: Pull-down  
11: Reserved
6.4.5  GPIO port input data register (GPIOx_IDR) (x = A to D, F)

Address offset: 0x10
Reset value: 0x0000 XXXX

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

ID15 ID14 ID13 ID12 ID11 ID10 ID9 ID8 ID7 ID6 ID5 ID4 ID3 ID2 ID1 ID0

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  ID[15:0]: Port x input data I/O pin y (y = 15 to 0)
These bits are read-only. They contain the input value of the corresponding I/O port.

6.4.6  GPIO port output data register (GPIOx_ODR) (x = A to D, F)

Address offset: 0x14
Reset value: 0x0000 0000

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

OD15 OD14 OD13 OD12 OD11 OD10 OD9 OD8 OD7 OD6 OD5 OD4 OD3 OD2 OD1 OD0

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  OD[15:0]: Port output data I/O pin y (y = 15 to 0)
These bits can be read and written by software.

Note:  For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A..D, F).

6.4.7  GPIO port bit set/reset register (GPIOx_BSRR) (x = A to D, F)

Address offset: 0x18
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS15 BS14 BS13 BS12 BS11 BS10 BS9 BS8 BS7 BS6 BS5 BS4 BS3 BS2 BS1 BS0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

w  w  w  w  w  w  w  w  w  w  w  w  w  w  w  w  w  w  w  w  w  w  w  w
6.4.8 GPIO port configuration lock register (GPIOx_LCKR) 
(x = A to D, F)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

**Note:** A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>BR[15:0]: Port x reset I/O pin y (y = 15 to 0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>These bits are write-only. A read to these bits returns the value 0x0000.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No action on the corresponding ODRx bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Resets the corresponding ODRx bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: If both BSx and BRx are set, BSx has priority.</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>BS[15:0]: Port x set I/O pin y (y = 15 to 0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>These bits are write-only. A read to these bits returns the value 0x0000.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No action on the corresponding ODRx bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Sets the corresponding ODRx bit</td>
<td></td>
</tr>
</tbody>
</table>
6.4.9 GPIO alternate function low register (GPIOx_AFRL)  
(x = A to D, F)

Address offset: 0x20

Reset value: 0x0000 0000

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

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

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

Bits 31:0 **AFSELy[3:0]**: Alternate function selection for port x pin y (y = 0..7)

These bits are written by software to configure alternate function I/Os

AFSELy selection:

0000: AF0  
0001: AF1  
0010: AF2  
0011: AF3  
0100: AF4  
0101: AF5  
0110: AF6  
0111: AF7  
1000: Reserved  
1001: Reserved  
1010: Reserved  
1011: Reserved  
1100: Reserved  
1101: Reserved  
1110: Reserved  
1111: Reserved

Bits 15:0 **LCK[15:0]**: Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is ‘0’.

0: Port configuration not locked  
1: Port configuration locked

**LCKK**: Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active  
1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

**LOCK** key write sequence:

WR LCKR[16] = ‘1’ + LCKR[15:0]  
WR LCKR[16] = ‘0’ + LCKR[15:0]  
WR LCKR[16] = ‘1’ + LCKR[15:0]

**RD LCKR**

RD LCKR[16] = ‘1’ (this read operation is optional but it confirms that the lock is active)

**Note**: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit returns ‘1’ until the next MCU reset or peripheral reset.

Bits 31:17 Reserved, must be kept at reset value.
### 6.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to D, F)

Address offset: 0x24
Reset value: 0x0000 0000

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

**Bits 31:0:** AFSELy[3:0]: Alternate function selection for port x pin y (y = 8..15)
These bits are written by software to configure alternate function I/Os

**AFSELy selection:**
- 0000: AF0
- 0001: AF1
- 0010: AF2
- 0011: AF3
- 0100: AF4
- 0101: AF5
- 0110: AF6
- 0111: AF7
- 1000: Reserved
- 1001: Reserved
- 1010: Reserved
- 1011: Reserved
- 1100: Reserved
- 1101: Reserved
- 1110: Reserved
- 1111: Reserved

### 6.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to D, F)

Address offset: 0x28
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>BR15</th>
<th>BR14</th>
<th>BR13</th>
<th>BR12</th>
<th>BR11</th>
<th>BR10</th>
<th>BR9</th>
<th>BR8</th>
<th>BR7</th>
<th>BR6</th>
<th>BR5</th>
<th>BR4</th>
<th>BR3</th>
<th>BR2</th>
<th>BR1</th>
<th>BR0</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

**Bits 31:16** Reserved, must be kept at reset value.

**Bits 15:0** BR[15:0]: Port x reset IO pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODx bit
1: Reset the corresponding ODx bit
## 6.4.12 GPIO register map

The following table gives the GPIO register map and reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>GPIOA_MODER</td>
<td>0x00</td>
<td>GPIOA_MODER</td>
</tr>
<tr>
<td></td>
<td>(where x = B..D, F)</td>
<td></td>
<td>(where x = B..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x04</td>
<td>GPIOA_OTYPER</td>
<td>0x04</td>
<td>GPIOA_OTYPER</td>
</tr>
<tr>
<td></td>
<td>(where x = A..D, F)</td>
<td></td>
<td>(where x = A..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0C</td>
<td>GPIOA_PUPDR</td>
<td>0x0C</td>
<td>GPIOA_PUPDR</td>
</tr>
<tr>
<td></td>
<td>(where x = B..D, F)</td>
<td></td>
<td>(where x = B..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x10</td>
<td>GPIOA_IDR</td>
<td>0x10</td>
<td>GPIOA_IDR</td>
</tr>
<tr>
<td></td>
<td>(where x = A..D, F)</td>
<td></td>
<td>(where x = A..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x14</td>
<td>GPIOA_ODR</td>
<td>0x14</td>
<td>GPIOA_ODR</td>
</tr>
<tr>
<td></td>
<td>(where x = A..D, F)</td>
<td></td>
<td>(where x = A..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x18</td>
<td>GPIOA_BSRR</td>
<td>0x18</td>
<td>GPIOA_BSRR</td>
</tr>
<tr>
<td></td>
<td>(where x = A..D, F)</td>
<td></td>
<td>(where x = A..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x1C</td>
<td>GPIOA_LCKR</td>
<td>0x1C</td>
<td>GPIOA_LCKR</td>
</tr>
<tr>
<td></td>
<td>(where x = A..D, F)</td>
<td></td>
<td>(where x = A..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
</tbody>
</table>

Table 33. GPIO register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>GPIOA_MODER</td>
<td>0x00</td>
<td>GPIOA_MODER</td>
</tr>
<tr>
<td></td>
<td>(where x = B..D, F)</td>
<td></td>
<td>(where x = B..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x04</td>
<td>GPIOA_OTYPER</td>
<td>0x04</td>
<td>GPIOA_OTYPER</td>
</tr>
<tr>
<td></td>
<td>(where x = A..D, F)</td>
<td></td>
<td>(where x = A..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0C</td>
<td>GPIOA_PUPDR</td>
<td>0x0C</td>
<td>GPIOA_PUPDR</td>
</tr>
<tr>
<td></td>
<td>(where x = B..D, F)</td>
<td></td>
<td>(where x = B..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x10</td>
<td>GPIOA_IDR</td>
<td>0x10</td>
<td>GPIOA_IDR</td>
</tr>
<tr>
<td></td>
<td>(where x = A..D, F)</td>
<td></td>
<td>(where x = A..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x14</td>
<td>GPIOA_ODR</td>
<td>0x14</td>
<td>GPIOA_ODR</td>
</tr>
<tr>
<td></td>
<td>(where x = A..D, F)</td>
<td></td>
<td>(where x = A..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x18</td>
<td>GPIOA_BSRR</td>
<td>0x18</td>
<td>GPIOA_BSRR</td>
</tr>
<tr>
<td></td>
<td>(where x = A..D, F)</td>
<td></td>
<td>(where x = A..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x1C</td>
<td>GPIOA_LCKR</td>
<td>0x1C</td>
<td>GPIOA_LCKR</td>
</tr>
<tr>
<td></td>
<td>(where x = A..D, F)</td>
<td></td>
<td>(where x = A..D, F)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
</tbody>
</table>
Refer to Section 2.2 on page 54 for the register boundary addresses.
7 System configuration controller (SYSCFG)

The devices feature a set of configuration registers. The main purposes of the system configuration controller are the following:

- Enabling/disabling I2C Fast Mode Plus on some I/O ports
- Enabling/disabling the analog switch booster
- Configuring the IR modulation signal and its output polarity
- Remapping of some I/O ports
- Remapping the memory located at the beginning of the code area
- Flag pending interrupts from each interrupt line
- Managing robustness feature

7.1 SYSCFG registers

7.1.1 SYSCFG configuration register 1 (SYSCFG_CFR1)

This register is used for specific configurations of memory and DMA requests remap and to control special I/O features.

Two bits are used to configure the type of memory accessible at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the hardware BOOT selection. After reset these bits take the value selected by the actual boot mode configuration.

Address offset: 0x00

Reset value: 0x0000 000X (X is the memory mode selected by the actual boot mode configuration)

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **I2C_PA10_FMP**: Fast Mode Plus (FM+) enable for PA10
   This bit is set and cleared by software. It enables I^2^C FM+ driving capability on PA10 I/O port.
   0: Disable
   1: Enable
   With this bit in disable state, the I^2^C FM+ driving capability on this I/O port can be enabled through one of I2Cx_FMP bits. When I^2^C FM+ is enabled, the speed control is ignored.

Bit 22 **I2C_PA9_FMP**: Fast Mode Plus (FM+) enable for PA9
   This bit is set and cleared by software. It enables I^2^C FM+ driving capability on PA9 I/O port.
   0: Disable
   1: Enable
   With this bit in disable state, the I^2^C FM+ driving capability on this I/O port can be enabled through one of I2Cx_FMP bits. When I^2^C FM+ is enabled, the speed control is ignored.

Bit 21 **I2C2_FMP**: Fast Mode Plus (FM+) enable for I2C2
   This bit is set and cleared by software. It enables I^2^C FM+ driving capability on I/O ports configured as I2C2 through GPIOx_AFR registers.
   0: Disable
   1: Enable
   With this bit in disable state, the I^2^C FM+ driving capability on I/O ports configured as I2C2 can be enabled through their corresponding I2Cx_FMP bit. When I^2^C FM+ is enabled, the speed control is ignored.

Bit 20 **I2C1_FMP**: Fast Mode Plus (FM+) enable for I2C1
   This bit is set and cleared by software. It enables I^2^C FM+ driving capability on I/O ports configured as I2C1 through GPIOx_AFR registers.
   0: Disable
   1: Enable
   With this bit in disable state, the I^2^C FM+ driving capability on I/O ports configured as I2C1 can be enabled through their corresponding I2Cx_FMP bit. When I^2^C FM+ is enabled, the speed control is ignored.

Bit 19 **I2C_PB9_FMP**: Fast Mode Plus (FM+) enable for PB9
   This bit is set and cleared by software. It enables I^2^C FM+ driving capability on PB9 I/O port.
   0: Disable
   1: Enable
   With this bit in disable state, the I^2^C FM+ driving capability on this I/O port can be enabled through one of I2Cx_FMP bits. When I^2^C FM+ is enabled, the speed control is ignored.

Bit 18 **I2C_PB8_FMP**: Fast Mode Plus (FM+) enable for PB8
   This bit is set and cleared by software. It enables I^2^C FM+ driving capability on PB8 I/O port.
   0: Disable
   1: Enable
   With this bit in disable state, the I^2^C FM+ driving capability on this I/O port can be enabled through one of I2Cx_FMP bits. When I^2^C FM+ is enabled, the speed control is ignored.

Bit 17 **I2C_PB7_FMP**: Fast Mode Plus (FM+) enable for PB7
   This bit is set and cleared by software. It enables I^2^C FM+ driving capability on PB7 I/O port.
   0: Disable
   1: Enable
   With this bit in disable state, the I^2^C FM+ driving capability on this I/O port can be enabled through one of I2Cx_FMP bits. When I^2^C FM+ is enabled, the speed control is ignored.
Bit 16 **I2C_PB6_FMP**: Fast Mode Plus (FM+) enable for PB6
This bit is set and cleared by software. It enables I^2^C FM+ driving capability on PB6 I/O port.
0: Disable
1: Enable
With this bit in disable state, the I^2^C FM+ driving capability on this I/O port can be enabled through one of I^2^Cx_FMP bits. When I^2^C FM+ is enabled, the speed control is ignored.

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

Bit 10 **UCPD2_STROBE**: UCPD2 pull-down configuration strobe
Upon power on, internal pull-down resistors on UCPD2 CC1 and CC2 PD0 and PD2 pins are enabled (connected).
The action of setting this bit has the following “strobing” effect:
- when UCPD2 is disabled: disable UCPD pull-down resistors on CC1 and CC2
- when UCPD2 is enabled, with CC1 and CC2 pin UCPD control bits configured: apply that configuration

See Section 35: USB Type-C™ / USB Power Delivery interface (UCPD) for details.
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 9 **UCPD1_STROBE**: UCPD1 pull-down configuration strobe
Upon power on, internal pull-down resistors on UCPD1 CC1 and CC2 PB15 and PA8 pins are enabled (connected).
The action of setting this bit has the following “strobing” effect:
- when UCPD1 is disabled: disable UCPD pull-down resistors on CC1 and CC2
- when UCPD1 is enabled, with CC1 and CC2 pin UCPD control bits configured: apply that configuration

See Section 35: USB Type-C™ / USB Power Delivery interface (UCPD) for details.
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 8 **BOOSTEN**: I/O analog switch voltage booster enable
This bit selects the way of supplying I/O analog switches:
0: VDD
1: Dedicated voltage booster (supplied by VDD)
When using the analog inputs, setting to 0 is recommended for high VDD, setting to 1 for low VDD (less than 2.4 V).

Bits 7:6 **IR_MOD[1:0]**: IR Modulation Envelope signal selection
This bitfield selects the signal for IR modulation envelope:
00: TIM16
01: USART1
10: USART4 on STM32G071xx and STM32G081xx, USART2 on STM32G031xx and STM32G041xx
11: Reserved

Bit 5 **IR_POL**: IR output polarity selection
0: Output of IRTIM (IR_OUT) is not inverted
1: Output of IRTIM (IR_OUT) is inverted
Bit 4 **PA12_RMP**: PA12 pin remapping
   This bit is set and cleared by software. When set, it remaps the PA12 pin to operate as PA10 GPIO port, instead as PA12 GPIO port.
   0: No remap (PA12)
   1: Remap (PA10)

Bit 3 **PA11_RMP**: PA11 pin remapping
   This bit is set and cleared by software. When set, it remaps the PA11 pin to operate as PA9 GPIO port, instead as PA11 GPIO port.
   0: No remap (PA11)
   1: Remap (PA9)

Bit 2 Reserved, must be kept at reset value.

**Bit 1:0 MEM_MODE[1:0]**: Memory mapping selection bits
   These bits are set and cleared by software. They control the memory internal mapping at address 0x0000 0000. After reset these bits take on the value selected by the actual boot mode configuration. Refer to Section 2.5: Boot configuration for more details.
   - x0: Main Flash memory mapped at 0x0000 0000
   - x1: System Flash memory mapped at 0x0000 0000
   - 11: Embedded SRAM mapped at 0x0000 0000

### 7.1.2 SYSCFG configuration register 2 (SYSCFG_CFGR2)

**Address offset**: 0x18

**System reset value**: 0x0000 0000

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

Bits 31:24 Reserved, must be kept at reset value

Bit 23 **PB2_CDEN**: PB2 clamping diode enable bit
   This bit is set and cleared by software. It enables (connects) a clamping diode to VDD on PB2 pin.
   0: Disable
   1: Enable

Bit 22 **PB1_CDEN**: PB1 clamping diode enable bit
   This bit is set and cleared by software. It enables (connects) a clamping diode to VDD on PB1 pin.
   0: Disable
   1: Enable
Bit 21 **PB0_CDEN**: PB0 clamping diode enable bit

This bit is set and cleared by software. It enables (connects) a clamping diode to VDD on PB0 pin.

0: Disable
1: Enable

Bit 20 **PA13_CDEN**: PA13 clamping diode enable bit

This bit is set and cleared by software. It enables (connects) a clamping diode to VDD on PA13 pin.

0: Disable
1: Enable

Bit 19 **PA6_CDEN**: PA6 clamping diode enable bit

This bit is set and cleared by software. It enables (connects) a clamping diode to VDD on PA6 pin.

0: Disable
1: Enable

Bit 18 **PA5_CDEN**: PA5 clamping diode enable bit

This bit is set and cleared by software. It enables (connects) a clamping diode to VDD on PA5 pin.

0: Disable
1: Enable

Bit 17 **PA3_CDEN**: PA3 clamping diode enable bit

This bit is set and cleared by software. It enables (connects) a clamping diode to VDD on PA3 pin.

0: Disable
1: Enable

Bit 16 **PA1_CDEN**: PA1 clamping diode enable bit

This bit is set and cleared by software. It enables (connects) a clamping diode to VDD on PA1 pin.

0: Disable
1: Enable

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

Bit 8 **SRAM_PEF**: SRAM parity error flag

This bit is set by hardware when an SRAM parity error is detected. It is cleared by software by writing 1.

0: No SRAM parity error detected
1: SRAM parity error detected

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ECC_LOCK**: ECC error lock bit

This bit is set by software and cleared by a system reset. It can be used to enable and lock the Flash ECC error signal connection to TIM1/15/16/17 Break input.

0: ECC error disconnected from TIM1/15/16/17 Break input
1: ECC error connected to TIM1/15/16/17 Break input
Bit 2  **PVD_LOCK**: PVD lock enable bit  
This bit is set by software and cleared by a system reset. It can be used to enable and lock  
the PVD connection to TIM1/15/16/17 Break input, as well as the PVDE and PLS[2:0] in the  
PWR_CR register.  
0: PVD interrupt disconnected from TIM1/15/16/17 Break input. PVDE and PLS[2:0] bits can  
be programmed by the application.  
1: PVD interrupt connected to TIM1/15/16/17 Break input, PVDE and PLS[2:0] bits are read  
only.

Bit 1  **SRAM_PARITY_LOCK**: SRAM parity lock bit  
This bit is set by software and cleared by a system reset. It can be used to enable and lock  
the SRAM parity error signal connection to TIM1/15/16/17 Break input.  
0: SRAM parity error disconnected from TIM1/15/16/17 Break input  
1: SRAM parity error connected to TIM1/15/16/17 Break input

Bit 0  **LOCKUP_LOCK**: Cortex®-M0+ LOCKUP bit enable bit  
This bit is set by software and cleared by a system reset. It can be used to enable and lock the  
connection of Cortex®-M0+ LOCKUP (Hardfault) output to TIM1/15/16/17 Break input.  
0: Cortex®-M0+ LOCKUP output disconnected from TIM1/15/16/17 Break input  
1: Cortex®-M0+ LOCKUP output connected to TIM1/15/16/17 Break input

**Note:** The bits 16 to 23 are only available in STM32G031xx and STM32G041xx. They are reserved in STM32G071xx and STM32G081xx.

### 7.1.3 SYSCFG interrupt line 0 status register (SYSCFG_ITLINE0)

A dedicated set of registers is implemented on the device to collect all pending interrupt  
sources associated with each interrupt line into a single register. This allows users to check by single read which peripheral requires service in case more than one source is associated to the interrupt line.

All bits in those registers are read only, set by hardware when there is corresponding interrupt request pending and cleared by resetting the interrupt source flags in the peripheral registers.

Address offset: 80h  
System reset value: 0x0000 0000

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

Bits 31:1 Reserved (read as 0)  

Bit 0  **WWDG**: Window watchdog interrupt pending flag

### 7.1.4 SYSCFG interrupt line 1 status register (SYSCFG_ITLINE1)

Address offset: 84h  
System reset value: 0x0000 0000
### RM0444 System configuration controller (SYSCFG)

#### 7.1.5 SYSCFG interrupt line 2 status register (SYSCFG_ITLINE2)

Address offset: 88h  
System reset value: 0x0000 0000

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

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

Bits 31:1 Reserved (read as 0)  
Bit 0 **PVDOUT**: PVD supply monitoring interrupt request pending ( EXTI line 16).

#### 7.1.6 SYSCFG interrupt line 3 status register (SYSCFG_ITLINE3)

Address offset: 8Ch  
System reset value: 0x0000 0000

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

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

Bits 31:3 Reserved (read as 0)  
Bit 1 **RTC**: RTC interrupt request pending ( EXTI line 19)  
Bit 0 **TAMP**: Tamper interrupt request pending ( EXTI line 21)

#### 7.1.7 SYSCFG interrupt line 4 status register (SYSCFG_ITLINE4)

Address offset: 90h

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

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

Bits 31:2 Reserved (read as 0)  
Bit 1 **FLASH_ECC**: Flash interface ECC interrupt request pending  
Bit 0 **FLASH_ITF**: Flash interface interrupt request pending
System reset value: 0x0000 0000

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

Bits 31:1 Reserved (read as 0)

Bit 0 **RCC**: Reset and clock control interrupt request pending
7.1.8 **SYSCFG interrupt line 5 status register (SYSCFG_ITLINE5)**

Address offset: 94h  
System reset value: 0x0000 0000

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

Bits 31:2 Reserved (read as 0)  
Bit 1 **EXTI1**: EXTI line 1 interrupt request pending  
Bit 0 **EXTI0**: EXTI line 0 interrupt request pending

7.1.9 **SYSCFG interrupt line 6 status register (SYSCFG_ITLINE6)**

Address offset: 98h  
System reset value: 0x0000 0000

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

Bits 31:2 Reserved (read as 0)  
Bit 1 **EXTI3**: EXTI line 3 interrupt request pending  
Bit 0 **EXTI2**: EXTI line 2 interrupt request pending

7.1.10 **SYSCFG interrupt line 7 status register (SYSCFG_ITLINE7)**

Address offset: 9Ch  
System reset value: 0x0000 0000

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

Bits 31:2 Reserved (read as 0)  
Bit 1 **EXTI15**: EXTI line 15 interrupt request pending  
Bit 0 **EXTI14**: EXTI line 14 interrupt request pending
### 7.1.11 SYSCFG interrupt line 8 status register (SYSCFG_ITLINE8)

Address offset: A0h

System reset value: 0x0000 0000

This register is only available on STM32G071xx and STM32G081xx.

<table>
<thead>
<tr>
<th>Bit 31:10</th>
<th>Reserved (read as 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 11</td>
<td>EXTI15: EXTI line 15 interrupt request pending</td>
</tr>
<tr>
<td>Bit 10</td>
<td>EXTI14: EXTI line 14 interrupt request pending</td>
</tr>
<tr>
<td>Bit 9</td>
<td>EXTI13: EXTI line 13 interrupt request pending</td>
</tr>
<tr>
<td>Bit 8</td>
<td>EXTI12: EXTI line 12 interrupt request pending</td>
</tr>
<tr>
<td>Bit 7</td>
<td>EXTI11: EXTI line 11 interrupt request pending</td>
</tr>
<tr>
<td>Bit 6</td>
<td>EXTI10: EXTI line 10 interrupt request pending</td>
</tr>
<tr>
<td>Bit 5</td>
<td>EXTI9: EXTI line 9 interrupt request pending</td>
</tr>
<tr>
<td>Bit 4</td>
<td>EXTI8: EXTI line 8 interrupt request pending</td>
</tr>
<tr>
<td>Bit 3</td>
<td>EXTI7: EXTI line 7 interrupt request pending</td>
</tr>
<tr>
<td>Bit 2</td>
<td>EXTI6: EXTI line 6 interrupt request pending</td>
</tr>
<tr>
<td>Bit 1</td>
<td>EXTI5: EXTI line 5 interrupt request pending</td>
</tr>
<tr>
<td>Bit 0</td>
<td>EXTI4: EXTI line 4 interrupt request pending</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31:2</th>
<th>Reserved (read as 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1</td>
<td>UCPD2: UCPD2 interrupt request pending (EXTI line 33)</td>
</tr>
<tr>
<td>Bit 0</td>
<td>UCPD1: UCPD1 interrupt request pending (EXTI line 32)</td>
</tr>
</tbody>
</table>

### 7.1.12 SYSCFG interrupt line 9 status register (SYSCFG_ITLINE9)

Address offset: A4h

System reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:10</th>
<th>Reserved (read as 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 31:16</td>
<td>Reserved (read as 0)</td>
</tr>
<tr>
<td>Bit 15</td>
<td>DMA1_CH1: DMA1 channel 1 interrupt request pending</td>
</tr>
</tbody>
</table>

222/1234 RM0444 Rev 2
7.1.13 **SYSCFG interrupt line 10 status register (SYSCFG_ITLINE10)**

Address offset: A8h

System reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Bit 23:16</th>
<th>Bit 15:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>DMA1_CH1</td>
<td>DMA1_CH2</td>
<td>DMA1_CH3</td>
<td>DMA1_CH4</td>
</tr>
</tbody>
</table>

Bits 31:1 Reserved (read as 0)

Bit 0 **DMA1_CH1**: DMA1 channel 1 interrupt request pending

This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bits 31:5 Reserved (read as 0)

Bit 4 **DMA1_CH7**: DMA1 channel 7 interrupt request pending

This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 3 **DMA1_CH6**: DMA1 channel 6 interrupt request pending

This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 2 **DMA1_CH5**: DMA1 channel 5 interrupt request pending

Bit 1 **DMA1_CH4**: DMA1 channel 4 interrupt request pending

Bit 0 **DMAMUX**: DMAMUX interrupt request pending

---

7.1.14 **SYSCFG interrupt line 11 status register (SYSCFG_ITLINE11)**

Address offset: ACh

System reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Bit 23:16</th>
<th>Bit 15:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>DMA1_CH7</td>
<td>DMA1_CH6</td>
<td>DMA1_CH5</td>
<td>DMA1_CH4</td>
</tr>
</tbody>
</table>

Bits 31:2 Reserved (read as 0)

Bit 1 **DMA1_CH3**: DMA1 channel 3 interrupt request pending

Bit 0 **DMA1_CH2**: DMA1 channel 2 interrupt request pending

Bits 31:5 Reserved (read as 0)

Bit 4 **DMA1_CH7**: DMA1 channel 7 interrupt request pending

This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 3 **DMA1_CH6**: DMA1 channel 6 interrupt request pending

This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 2 **DMA1_CH5**: DMA1 channel 5 interrupt request pending

Bit 1 **DMA1_CH4**: DMA1 channel 4 interrupt request pending

Bit 0 **DMAMUX**: DMAMUX interrupt request pending
7.1.15 SYSCFG interrupt line 12 status register (SYSCFG_ITLINE12)

Address offset: B0h
System reset value: 0x0000 0000

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

Bits 31:3 Reserved (read as 0)

- Bit 2 COMP2: Comparator 2 interrupt request pending (EXTI line 18)
  This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

- Bit 1 COMP1: Comparator 1 interrupt request pending (EXTI line 17)
  This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

- Bit 0 ADC: ADC interrupt request pending

7.1.16 SYSCFG interrupt line 13 status register (SYSCFG_ITLINE13)

Address offset: B4h
System reset value: 0x0000 0000

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

Bits 31:4 Reserved (read as 0)

- Bit 3 TIM1_BRK: Timer 1 break interrupt request pending
- Bit 2 TIM1_UPD: Timer 1 update interrupt request pending
- Bit 1 TIM1_TRG: Timer 1 trigger interrupt request pending
- Bit 0 TIM1_CCU: Timer 1 commutation interrupt request pending
### 7.1.17 SYS CFG interrupt line 14 status register (SYS CFG ITLINE14)

Address offset: B8h

System reset value: 0x0000 0000

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

Bits 31:1 Reserved (read as 0)

Bit 0 **TIM1 CC**: Timer 1 capture compare interrupt request pending

### 7.1.18 SYS CFG interrupt line 15 status register (SYS CFG ITLINE15)

Address offset: BCh

System reset value: 0x0000 0000

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

Bits 31:1 Reserved (read as 0)

Bit 0 **TIM2**: Timer 2 interrupt request pending

### 7.1.19 SYS CFG interrupt line 16 status register (SYS CFG ITLINE16)

Address offset: C0h

System reset value: 0x0000 0000

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

Bits 31:1 Reserved (read as 0)

Bit 0 **TIM3**: Timer 3 interrupt request pending
### 7.1.20 SYSCFG interrupt line 17 status register (SYSCFG_ITLINE17)

Address offset: C4h  
System reset value: 0x0000 0000

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

Bits 31:3 Reserved (read as 0)  
Bit 2 LPTIM1: Low-power timer 1 interrupt request pending (EXTI line 29)  
Bit 1 DAC: DAC underrun interrupt request pending  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.  
Bit 0 TIM6: Timer 6 interrupt request pending  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

### 7.1.21 SYSCFG interrupt line 18 status register (SYSCFG_ITLINE18)

Address offset: C8h  
System reset value: 0x0000 0000

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

Bits 31:2 Reserved (read as 0)  
Bit 1 LPTIM2: Low-power timer 2 interrupt request pending (EXTI line 30)  
Bit 0 TIM7: Timer 7 interrupt request pending  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

### 7.1.22 SYSCFG interrupt line 19 status register (SYSCFG_ITLINE19)

Address offset: CCh  
System reset value: 0x0000 0000

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

226/1234  
RM0444 Rev 2
### System configuration controller (SYSCFG)

**7.1.23 SYSCFG interrupt line 20 status register (SYSCFG_ITLINE20)**

Address offset: D0h  
System reset value: 0x0000 0000

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

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

Bits 31:1 Reserved (read as 0)  
Bit 0 **TIM14**: Timer 14 interrupt request pending

**7.1.24 SYSCFG interrupt line 21 status register (SYSCFG_ITLINE21)**

Address offset: D4h  
System reset value: 0x0000 0000

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

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

Bits 31:1 Reserved (read as 0)  
Bit 0 **TIM15**: Timer 15 interrupt request pending  
This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

**7.1.25 SYSCFG interrupt line 22 status register (SYSCFG_ITLINE22)**

Address offset: D8h  
System reset value: 0x0000 0000

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

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

Bits 31:1 Reserved (read as 0)  
Bit 0 **TIM16**: Timer 16 interrupt request pending
### 7.1.26 SYSCFG interrupt line 23 status register (SYSCFG_ITLINE23)

Address offset: DCh

System reset value: 0x0000 0000

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

Bits 31:1 Reserved (read as 0)

Bit 0 **TIM17**: Timer 17 interrupt request pending

### 7.1.27 SYSCFG interrupt line 24 status register (SYSCFG_ITLINE24)

Address offset: E0h

System reset value: 0x0000 0000

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

Bits 31:1 Reserved (read as 0)

Bit 0 **I2C1**: I2C1 interrupt request pending, combined with EXTI line 23

### 7.1.28 SYSCFG interrupt line 25 status register (SYSCFG_ITLINE25)

Address offset: E4h

System reset value: 0x0000 0000

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

Bits 31:1 Reserved (read as 0)

Bit 0 **I2C2**: I2C2 interrupt request pending
7.1.29 SYSCFG interrupt line 26 status register (SYSCFG_ITLINE26)

Address offset: E8h

System reset value: 0x0000 0000

Bits 31:1 Reserved (read as 0)

Bit 0  **SPI1**: SPI1 interrupt request pending

7.1.30 SYSCFG interrupt line 27 status register (SYSCFG_ITLINE27)

Address offset: EC

System reset value: 0x0000 0000

Bits 31:1 Reserved (read as 0)

Bit 0  **SPI2**: SPI2 interrupt request pending

7.1.31 SYSCFG interrupt line 28 status register (SYSCFG_ITLINE28)

Address offset: F0h

System reset value: 0x0000 0000
7.1.32 SYSCFG interrupt line 29 status register (SYSCFG_ITLINE29)

Address offset: F4h

System reset value: 0x0000 0000

Bits 31:1 Reserved (read as 0)

Bit 0 **USART2**: USART2 interrupt request pending, combined with EXTI line 26

7.1.33 SYSCFG interrupt line 30 status register (SYSCFG_ITLINE30)

Address offset: F8h

System reset value: 0x0000 0000

Bits 31:3 Reserved (read as 0)

Bit 2 **LPUART1**: LPUART1 interrupt request pending (EXTI line 28)

Bit 1 **USART4**: USART4 interrupt request pending
   
   This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 0 **USART3**: USART3 interrupt request pending, combined with EXTI line 28.
   
   This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

7.1.34 SYSCFG interrupt line 31 status register (SYSCFG_ITLINE31)

Address offset: FCh

System reset value: 0x0000 0000

Bits 31:1 Reserved (read as 0)

Bit 0 **CEC**: CEC interrupt request pending, combined with EXTI line 27

   This bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.
7.1.34 SYSCFG interrupt line 31 status register (SYSCFG_ITLINE31)

Address offset: FCh
System reset value: 0x0000 0000

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

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

Bits 31:2 Reserved (read as 0)
- Bit 1 **AES**: AES interrupt request pending
- Bit 0 **RNG**: RNG interrupt request pending

7.1.35 SYSCFG register map
The following table gives the SYSCFG register map and the reset values.

**Table 34. SYSCFG register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>SYSCFG_CFGR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

0x04 to 0x17
Reserved

0x18
SYSCFG_CFGR2
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

0x1D to 0x7F
Reserved

0x80
SYSCFG_ITLINE0
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

0x84
SYSCFG_ITLINE1
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

0x88
SYSCFG_ITLINE2
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

STI
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x8C</td>
<td>SYSCFG_ITLINE3</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x90</td>
<td>SYSCFG_ITLINE4</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x94</td>
<td>SYSCFG_ITLINE5</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x98</td>
<td>SYSCFG_ITLINE6</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x9C</td>
<td>SYSCFG_ITLINE7</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0xA0</td>
<td>SYSCFG_ITLINE8</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0xA4</td>
<td>SYSCFG_ITLINE9</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0xA8</td>
<td>SYSCFG_ITLINE10</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0xAC</td>
<td>SYSCFG_ITLINE11</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0xB0</td>
<td>SYSCFG_ITLINE12</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0xB4</td>
<td>SYSCFG_ITLINE13</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0xB8</td>
<td>SYSCFG_ITLINE14</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0xBC</td>
<td>SYSCFG_ITLINE15</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0xC0</td>
<td>SYSCFG_ITLINE16</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
</tr>
</tbody>
</table>
Table 34. SYSCFG register map and reset values (continued)

| Offset | Register         | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xC4   | SYSCFG_ITLINE17  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xC8   | SYSCFG_ITLINE18  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xCC   | SYSCFG_ITLINE19  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xD0   | SYSCFG_ITLINE20  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xD4   | SYSCFG_ITLINE21  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xD8   | SYSCFG_ITLINE22  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xDC   | SYSCFG_ITLINE23  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xE0   | SYSCFG_ITLINE24  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xE4   | SYSCFG_ITLINE25  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xE8   | SYSCFG_ITLINE26  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xEC   | SYSCFG_ITLINE27  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xF0   | SYSCFG_ITLINE28  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xF4   | SYSCFG_ITLINE29  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xF8   | SYSCFG_ITLINE30  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFC   | SYSCFG_ITLINE31  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.2 on page 54 for the register boundary addresses.
8  Interconnect matrix

8.1  Introduction

Several peripherals have direct connections between them.
This allows autonomous communication and/or synchronization between peripherals, saving CPU resources thus power consumption.
In addition, these hardware connections remove software latency and allow design of predictable systems.
Depending on peripherals, these interconnections can operate in Run, Sleep, Low-power run, Low-power sleep, Stop 0, and Stop 1 modes.

8.2  Connection summary

<table>
<thead>
<tr>
<th>Source</th>
<th>TIM1</th>
<th>TIM2</th>
<th>TIM3</th>
<th>TIM4</th>
<th>TIM15</th>
<th>TIM16</th>
<th>TIM17</th>
<th>LPTIM1</th>
<th>LPTIM2</th>
<th>ADC</th>
<th>DAC</th>
<th>DMAUX</th>
<th>COMP1</th>
<th>COMP2</th>
<th>IRTIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM1</td>
<td></td>
<td>8.3.1</td>
<td>8.3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM2</td>
<td>8.3.1</td>
<td></td>
<td>8.3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM3</td>
<td>8.3.1</td>
<td>8.3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM14</td>
<td></td>
<td>8.3.1</td>
<td>8.3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM15</td>
<td>8.3.1</td>
<td>8.3.1</td>
<td>8.3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM16</td>
<td></td>
<td></td>
<td></td>
<td>8.3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM17</td>
<td>8.3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPTIM1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPTIM2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USART1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.3.11</td>
</tr>
<tr>
<td>USART4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.3.11</td>
<td></td>
</tr>
<tr>
<td>ADC</td>
<td>8.3.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T. sensor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VBAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREFINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSE</td>
<td></td>
<td>8.3.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.3.5</td>
<td></td>
</tr>
</tbody>
</table>

Table 35. Interconnect matrix\(^{(1)(2)}\)
8.3 Interconnection details

8.3.1 From TIM1, TIM2, TIM3, TIM15, TIM16, and TIM17, to TIM1, TIM2, TIM3, and TIM15

Purpose

Some of the TIMx timers are linked together internally for timer synchronization or chaining.

When one timer is configured in master mode, it can reset, start, stop or clock the counter of another timer configured in slave mode.

A description of the feature is provided in: Section 21.3.19: Timer synchronization.

The modes of synchronization are detailed in:
- Section 20.3.26: Timer synchronization for advanced-control timer TIM1
- Section 21.3.18: Timers and external trigger synchronization for general-purpose timers TIM2/TIM3
- Section 24.4.19: External trigger synchronization (TIM15 only) for general-purpose timer TIM15

Triggering signals

The output (from master) is on signal TIMx_TRGO (and TIMx_TRGOx), following a configurable timer event.

With TIM14, TIM16, and TIM17 timers that do not have a trigger output, the output compare 1 is used instead.

The input (to slave) is on signals TIMx_ITR0/ITR1/ITR2/ITR3.
The input and output signals for TIM1 are shown in Figure 103: Advanced-control timer block diagram.

The possible master/slave connections are given in Table 109: TIMx internal trigger connection and Table 116: TIMx Internal trigger connection.

Relevant power modes
These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

8.3.2 From TIM1, TIM2, TIM3, TIM6, TIM15, and EXTI, to ADC

Purpose
The general-purpose timers TIM2, TIM3 and TIM15, basic timer TIM6, advanced-control timer TIM1, and EXTI can be used to generate an ADC triggering event.

TIMx synchronization is described in: Section 20.3.27: ADC synchronization.

ADC synchronization is described in: Section 14.4: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN).

Triggering signals
The output (from timer) is on signal TIMx_TRGO, TIMx_TRGO2 or TIMx_CCx event.
The input (to ADC) is on signal EXT[15:0], JEXT[15:0].
The connection between timers and ADC is provided in Table 62: External triggers.

Relevant power modes
These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

8.3.3 From ADC to TIM1

Purpose
ADC can provide trigger event through watchdog signals to the advanced-control timer TIM1.

A description of the ADC analog watchdog setting is provided in: Section 14.7: Analog window watchdog (AWD1EN, AWD1SGL, AWD1CH, ADC_AWDxCR, ADC_AWDxTR).

Trigger settings on the timer are provided in: Section 20.3.4: External trigger input.

Triggering signals
The output (from ADC) is on signals ADCn_AWDx_OUT n = 1, 2, 3 (for ADC x = 1, 2, 3 (three watchdogs per ADC) and the input (to timer) on signal TIMx_ETR (external trigger).

Relevant power modes
This interconnection operates in Run, Sleep, Low-power run, and Low-power sleep power modes.
8.3.4 From TIM1, TIM2, TIM3, TIM6, TIM7, TIM15, LPTIM1, LPTIM2, and EXTI, to DAC

**Purpose**

General-purpose timers TIM2/TIM3/TIM15, basic timers TIM6, TIM7, low-power timers LPTIM1/LPTIM2, and advanced control timer TIM1 can trigger a DAC conversion.

**Triggering signals**

The TIMx_TRGO output of each timer is directly connected to corresponding DAC input. Selection of DAC triggering input is provided in *Section 15.4.7: DAC trigger selection* (single and dual mode).

**Relevant power modes**

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

8.3.5 From HSE, LSE, LSI, MCO, RTC and TAMP, to TIM2, TIM14, TIM16, and TIM17

**Purpose**

External clocks (HSE, LSE), internal clock (LSI), microcontroller output clock (MCO), RTC clock, RTC wakeup interrupt, and GPIO can be selected as inputs to capture channel 1 of some of TIM14/16/TIM17 timers.

The timers allow calibrating or precisely measuring internal clocks such as HSI16 or LSI, using accurate clocks such as LSE or HSE/32 for timing reference. See details in *Section 5.2.15: Internal/external clock measurement with TIM14/TIM16/TIM17*.

When low-speed external (LSE) oscillator is used, no additional hardware connections are required.

External clock LSE can be used as input to general-purpose timers (TIM2) on TIM2_ETR input, see *Section 21.4.24: TIM2 alternate function option register 1 (TIM2_AF1)*.

**Relevant power modes**

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

8.3.6 From RTC, TAMP, COMP1, and COMP2, to LPTIM1 and LPTIM2

**Purpose**

RTC alarm A/B, TAMP1/2 input detection, COMP1/2_OUT and GPIO alternate function can be used as trigger to start LPTIM counters LPTIM1/2.

**Triggering signals**

This trigger feature is described in *Section 25.4.7: Trigger multiplexer* (and following sections).
The input selection is described in Table 124: LPTIM1 external trigger connection and Table 125: LPTIM2 external trigger connection.

**Relevant power modes**

These interconnections operate in Run, Sleep, Low-power run, Low-power sleep, Stop 0, and Stop 1 power modes.

### 8.3.7 From TIM1, TIM2, TIM3, and TIM15, to COMP1 and COMP2

**Purpose**

Advanced-control timer TIM1 and general-purpose timers TIM2, TIM3, and TIM15 can be used as blanking window input to COMP1 and COMP2.

The blanking function is described in Section 17.3.7: Comparator output blanking function.

The blanking sources are given in:
- Section 17.6.1: Comparator 1 control and status register (COMP1_CSR) bits 20:18 BLANKING[2:0]
- Section 17.6.2: Comparator 2 control and status register (COMP2_CSR) bits 20:18 BLANKING[2:0]

**Triggering signals**

Timer output signal TIMx_OCx are the inputs to blanking source of COMP1/COMP2.

**Relevant power modes**

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

### 8.3.8 From internal analog sources to ADC

**Purpose**

Internal temperature sensor output voltage $V_{TS}$, internal reference voltage $V_{REFINT}$ and $V_{BAT}$ monitoring channel are connected to ADC input channels.

More information is in:
- Section 14.2: ADC main features
- Section 14.3.8: Channel selection (CHSEL, SCANDIR, CHSELRMOD)
- Figure 14.9: Temperature sensor and internal reference voltage
- Figure 14.10: Battery voltage monitoring

**Relevant power modes**

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.
8.3.9 From COMP1 and COMP2, to TIM1, TIM2, TIM3, TIM15, TIM16, and TIM17

Purpose
COMP1 and COMP2 comparator outputs can be connected to input capture or TIMx_ETR inputs of TIM1, TIM2, or TIM3.

The connection to ETR is described in Section 20.3.4: External trigger input.

COMP1 and COMP2 comparator outputs can also act as TIMx_BKIN or TIMx_BKIN2 break input signals for TIM1, TIM15, TIM16, and TIM17, through selecting GPIO alternate function using open drain connection of I/O. See Section 20.3.17: Bidirectional break inputs.

The possible connections are given in:
- Section 20.4.23: TIM1 option register 1 (TIM1_OR1)
- Section 20.4.28: TIM1 Alternate function register 2 (TIM1_AF2)
- Section 21.4.22: TIM2 option register 1 (TIM2_OR1)
- Section 24.3: TIM16/TIM17 main features

Relevant power modes
These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

8.3.10 From system errors to TIM1, TIM2, TIM3, TIM15, TIM16, and TIM17

Purpose
CSS, CPU hardfault, RAM parity error, FLASH ECC double error detection, PVD can generate system errors in the form of timer break toward TIM1, TIM2, TIM3, TIM15, TIM16, and TIM17.

The purpose of the break function is to protect power switches driven by PWM signals from the timers.

List of possible source of break are described in:
- Section 20.3.16: Using the break function (TIM1)
- Section 24.4.13: Using the break function (TIM15/TIM16/TIM17)
- Figure 237: TIM15 block diagram
- Figure 238: TIM16/TIM17 block diagram

Relevant power modes
These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

8.3.11 From TIM16, TIM17, USART1, and USART4, to IRTIM

Purpose
TIMx_OC1 output channel of TIM16 or TIM17 timers, associated with USART1 or USART4 transmission signal, can generate the infrared output waveform.

The functionality is described in Section 26: Infrared interface (IRTIM).
Relevant power modes
These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

8.3.12 From TIM14, LPTIM1, and LPTIM2, to DMAMUX

Purpose
TIM14 general-purpose timer, LPTIM1 and LPTIM2 low-power timers, and EXTI, can be used as triggering event to DMAMUX.

Relevant power modes
These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.
9 Direct memory access controller (DMA)

9.1 Introduction

The direct memory access (DMA) controller is a bus master and system peripheral. The DMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories, upon the control of an off-loaded CPU.

The DMA controller features a single AHB master architecture.

There is one instance of DMA, with 7 channels on STM32G071xx and STM32G081xx devices and 5 on STM32G031xx and STM32G041xx.

Each channel is dedicated to managing memory access requests from one or more peripherals. The DMA includes an arbiter for handling the priority between DMA requests.

9.2 DMA main features

- Single AHB master
- Peripheral-to-memory, memory-to-peripheral, memory-to-memory and peripheral-to-peripheral data transfers
- Access, as source and destination, to on-chip memory-mapped devices such as Flash memory, SRAM, and AHB and APB peripherals
- All DMA channels independently configurable:
  - Each channel is associated either with a DMA request signal coming from a peripheral, or with a software trigger in memory-to-memory transfers. This configuration is done by software.
  - Priority between the requests is programmable by software (4 levels per channel: very high, high, medium, low) and by hardware in case of equality (such as request to channel 1 has priority over request to channel 2).
  - Transfer size of source and destination are independent (byte, half-word, word), emulating packing and unpacking. Source and destination addresses must be aligned on the data size.
  - Support of transfers from/to peripherals to/from memory with circular buffer management
  - Programmable number of data to be transferred: 0 to \(2^{16} - 1\)
- Generation of an interrupt request per channel. Each interrupt request is caused from any of the three DMA events: transfer complete, half transfer, or transfer error.
9.3 DMA implementation

9.3.1 DMA

DMA is implemented with the hardware configuration parameters shown in Table 36.

Table 36. DMA implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>DMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels</td>
<td>7/5 (1)</td>
</tr>
</tbody>
</table>

1. Seven channels for STM32G071xx and STM32G081xx, five for STM32G031xx and STM32G041xx.

9.3.2 DMA request mapping

The DMA controller is connected to DMA requests from the AHB/APB peripherals through the DMAMUX peripheral.

For the mapping of the different requests, refer to the Section 10.3: DMAMUX implementation.

9.4 DMA functional description

9.4.1 DMA block diagram

The DMA block diagram is shown in Figure 20.

Figure 20. DMA block diagram
The DMA controller performs direct memory transfer by sharing the AHB system bus with other system masters. The bus matrix implements round-robin scheduling. DMA requests may stop the CPU access to the system bus for a number of bus cycles, when CPU and DMA target the same destination (memory or peripheral).

According to its configuration through the AHB slave interface, the DMA controller arbitrates between the DMA channels and their associated received requests. The DMA controller also schedules the DMA data transfers over the single AHB port master.

The DMA controller generates an interrupt per channel to the interrupt controller.

### 9.4.2 DMA pins and internal signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dma_req[x]</td>
<td>Input</td>
<td>DMA channel x request</td>
</tr>
<tr>
<td>dma_ack[x]</td>
<td>Output</td>
<td>DMA channel x acknowledge</td>
</tr>
<tr>
<td>dma_it[x]</td>
<td>Output</td>
<td>DMA channel x interrupt</td>
</tr>
</tbody>
</table>

### 9.4.3 DMA transfers

The software configures the DMA controller at channel level, in order to perform a block transfer, composed of a sequence of AHB bus transfers.

A DMA block transfer may be requested from a peripheral, or triggered by the software in case of memory-to-memory transfer.

After an event, the following steps of a single DMA transfer occur:

1. The peripheral sends a single DMA request signal to the DMA controller.
2. The DMA controller serves the request, depending on the priority of the channel associated to this peripheral request.
3. As soon as the DMA controller grants the peripheral, an acknowledge is sent to the peripheral by the DMA controller.
4. The peripheral releases its request as soon as it gets the acknowledge from the DMA controller.
5. Once the request is de-asserted by the peripheral, the DMA controller releases the acknowledge.

The peripheral may order a further single request and initiate another single DMA transfer.

The request/acknowledge protocol is used when a peripheral is either the source or the destination of the transfer. For example, in case of memory-to-peripheral transfer, the peripheral initiates the transfer by driving its single request signal to the DMA controller. The DMA controller reads then a single data in the memory and writes this data to the peripheral.

For a given channel x, a DMA block transfer consists of a repeated sequence of:

- a single DMA transfer, encapsulating two AHB transfers of a single data, over the DMA AHB bus master:
  - a single data read (byte, half-word or word) from the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.
The start address used for the first single transfer is the base address of the peripheral or memory, and is programmed in the DMA_CPARx or DMA_CMARx register.

- a single data write (byte, half-word or word) to the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.

The start address used for the first transfer is the base address of the peripheral or memory, and is programmed in the DMA_CPARx or DMA_CMARx register.

- post-decrementing of the programmed DMA_CNDTRx register
  This register contains the remaining number of data items to transfer (number of AHB ‘read followed by write’ transfers).

This sequence is repeated until DMA_CNDTRx is null.

Note: The AHB master bus source/destination address must be aligned with the programmed size of the transferred single data to the source/destination.

9.4.4 DMA arbitration

The DMA arbiter manages the priority between the different channels.

When an active channel x is granted by the arbiter (hardware requested or software triggered), a single DMA transfer is issued (such as a AHB ‘read followed by write’ transfer of a single data). Then, the arbiter considers again the set of active channels and selects the one with the highest priority.

The priorities are managed in two stages:

- software: priority of each channel is configured in the DMA_CCRx register, to one of the four different levels:
  - very high
  - high
  - medium
  - low

- hardware: if two requests have the same software priority level, the channel with the lowest index gets priority. For example, channel 2 gets priority over channel 4.

When a channel x is programmed for a block transfer in memory-to-memory mode, re arbitration is considered between each single DMA transfer of this channel x. Whenever there is another concurrent active requested channel, the DMA arbiter automatically alternates and grants the other highest-priority requested channel, which may be of lower priority than the memory-to-memory channel.

9.4.5 DMA channels

Each channel may handle a DMA transfer between a peripheral register located at a fixed address, and a memory address. The amount of data items to transfer is programmable. The register that contains the amount of data items to transfer is decremented after each transfer.

A DMA channel is programmed at block transfer level.
Programmable data sizes

The transfer sizes of a single data (byte, half-word, or word) to the peripheral and memory are programmable through, respectively, the PSIZE[1:0] and MSIZE[1:0] fields of the DMA_CCRx register.

Pointer incrementation

The peripheral and memory pointers may be automatically incremented after each transfer, depending on the PINC and MINC bits of the DMA_CCRx register.

If the incremented mode is enabled (PINC or MINC set to 1), the address of the next transfer is the address of the previous one incremented by 1, 2 or 4, depending on the data size defined in PSIZE[1:0] or MSIZE[1:0]. The first transfer address is the one programmed in the DMA_CPARx or DMA_CMARx register. During transfers, these registers keep the initially programmed value. The current transfer addresses (in the current internal peripheral/memory address address register) are not accessible by software.

If the channel x is configured in non-circular mode, no DMA request is served after the last data transfer (once the number of single data to transfer reaches zero). The DMA channel must be disabled in order to reload a new number of data items into the DMA_CNDTRx register.

Note: If the channel x is disabled, the DMA registers are not reset. The DMA channel registers (DMA_CCRx, DMA_CPARx and DMA_CMARx) retain the initial values programmed during the channel configuration phase.

In circular mode, after the last data transfer, the DMA_CNDTRx register is automatically reloaded with the initially programmed value. The current internal address registers are reloaded with the base address values from the DMA_CPARx and DMA_CMARx registers.

Channel configuration procedure

The following sequence is needed to configure a DMA channel x:

1. Set the peripheral register address in the DMA_CPARx register. The data is moved from/to this address to/from the memory after the peripheral event, or after the channel is enabled in memory-to-memory mode.
2. Set the memory address in the DMA_CMARx register. The data is written to/read from the memory after the peripheral event or after the channel is enabled in memory-to-memory mode.
3. Configure the total number of data to transfer in the DMA_CNDTRx register. After each data transfer, this value is decremented.
4. Configure the parameters listed below in the DMA_CCRx register:
   - the channel priority
   - the data transfer direction
   - the circular mode
   - the peripheral and memory incremented mode
   - the peripheral and memory data size
   - the interrupt enable at half and/or full transfer and/or transfer error
5. Activate the channel by setting the EN bit in the DMA_CCRx register.

A channel, as soon as enabled, may serve any DMA request from the peripheral connected to this channel, or may start a memory-to-memory block transfer.
Direct memory access controller (DMA)

Note: The two last steps of the channel configuration procedure may be merged into a single access to the DMA_CCRx register, to configure and enable the channel.

Channel state and disabling a channel

A channel x in active state is an enabled channel (read DMA_CCRx.EN = 1). An active channel x is a channel that must have been enabled by the software (DMA_CCRx.EN set to 1) and afterwards with no occurred transfer error (DMA_ISR.TEIFx = 0). In case there is a transfer error, the channel is automatically disabled by hardware (DMA_CCRx.EN = 0).

The three following use cases may happen:

- **Suspend and resume a channel**
  - This corresponds to the two following actions:
    - An active channel is disabled by software (writing DMA_CCRx.EN = 0 whereas DMA_CCRx.EN = 1).
    - The software enables the channel again (DMA_CCRx.EN set to 1) without reconfiguring the other channel registers (such as DMA_CNDTRx, DMA_CPARx and DMA_CMARx).
  - This case is not supported by the DMA hardware, that does not guarantee that the remaining data transfers are performed correctly.

- **Stop and abort a channel**
  - If the application does not need any more the channel, this active channel can be disabled by software. The channel is stopped and aborted but the DMA_CNDTRx register content may not correctly reflect the remaining data transfers versus the aborted source and destination buffer/register.

- **Abort and restart a channel**
  - This corresponds to the software sequence: disable an active channel, then reconfigure the channel and enable it again.
  - This is supported by the hardware if the following conditions are met:
    - The application guarantees that, when the software is disabling the channel, a DMA data transfer is not occurring at the same time over its master port. For example, the application can first disable the peripheral in DMA mode, in order to ensure that there is no pending hardware DMA request from this peripheral.
    - The software must operate separated write accesses to the same DMA_CCRx register: First disable the channel. Second reconfigure the channel for a next block transfer including the DMA_CCRx if a configuration change is needed. There are read-only DMA_CCRx register fields when DMA_CCRx.EN=1. Finally enable again the channel.

  When a channel transfer error occurs, the EN bit of the DMA_CCRx register is cleared by hardware. This EN bit can not be set again by software to re-activate the channel x, until the TEIFx bit of the DMA_ISR register is set.

Circular mode (in memory-to-peripheral/peripheral-to-memory transfers)

The circular mode is available to handle circular buffers and continuous data flows (such as ADC scan mode). This feature is enabled using the CIRC bit in the DMA_CCRx register.

Note: The circular mode must not be used in memory-to-memory mode. Before enabling a channel in circular mode (CIRC = 1), the software must clear the MEM2MEM bit of the DMA_CCRx register. When the circular mode is activated, the amount of data to transfer is
automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

In order to stop a circular transfer, the software needs to stop the peripheral from generating DMA requests (such as quit the ADC scan mode), before disabling the DMA channel. The software must explicitly program the DMA_CNDTRx value before starting/enabling a transfer, and after having stopped a circular transfer.

**Memory-to-memory mode**

The DMA channels may operate without being triggered by a request from a peripheral. This mode is called memory-to-memory mode, and is initiated by software.

If the MEM2MEM bit in the DMA_CCRx register is set, the channel, if enabled, initiates transfers. The transfer stops once the DMA_CNDTRx register reaches zero.

*Note:* The memory-to-memory mode must not be used in circular mode. Before enabling a channel in memory-to-memory mode (MEM2MEM = 1), the software must clear the CIRC bit of the DMA_CCRx register.

**Peripheral-to-peripheral mode**

Any DMA channel can operate in peripheral-to-peripheral mode:

- when the hardware request from a peripheral is selected to trigger the DMA channel
  
  This peripheral is the DMA initiator and paces the data transfer from/to this peripheral to/from a register belonging to another memory-mapped peripheral (this one being not configured in DMA mode).

- when no peripheral request is selected and connected to the DMA channel
  
  The software configures a register-to-register transfer by setting the MEM2MEM bit of the DMA_CCRx register.

**Programming transfer direction, assigning source/destination**

The value of the DIR bit of the DMA_CCRx register sets the direction of the transfer, and consequently, it identifies the source and the destination, regardless the source/destination type (peripheral or memory):

- **DIR = 1** defines typically a memory-to-peripheral transfer. More generally, if DIR = 1:
  
  - The source attributes are defined by the DMA_MARx register, the MSIZE[1:0] field and MINC bit of the DMA_CCRx register.
    
    Regardless of their usual naming, these ‘memory’ register, field and bit are used to define the source peripheral in peripheral-to-peripheral mode.

  - The destination attributes are defined by the DMA_PARx register, the PSIZE[1:0] field and PINC bit of the DMA_CCRx register.
    
    Regardless of their usual naming, these ‘peripheral’ register, field and bit are used to define the destination memory in memory-to-memory mode.

- **DIR = 0** defines typically a peripheral-to-memory transfer. More generally, if DIR = 0:
  
  - The source attributes are defined by the DMA_PARx register, the PSIZE[1:0] field and PINC bit of the DMA_CCRx register.
    
    Regardless of their usual naming, these ‘peripheral’ register, field and bit are used to define the source memory in memory-to-memory mode.

  - The destination attributes are defined by the DMA_MARx register, the MSIZE[1:0] field and MINC bit of the DMA_CCRx register.
Regardless of their usual naming, these ‘memory’ register, field and bit are used to define the destination peripheral in peripheral-to-peripheral mode.
### 9.4.6 DMA data width, alignment and endianness

When PSIZE[1:0] and MSIZE[1:0] are not equal, the DMA controller performs some data alignments as described in **Table 38**.

**Table 38. Programmable data width and endian behavior (when PINC = MINC = 1)**

<table>
<thead>
<tr>
<th>Source port width (MSIZE if DIR = 1, else PSIZE)</th>
<th>Destinations port width (PSIZE if DIR = 1, else MSIZE)</th>
<th>Number of data items to transfer (NDT)</th>
<th>Source content: address / data (DMA_CMARx if DIR = 1, else DMA_CPARx)</th>
<th>DMA transfers</th>
<th>Destination content: address / data (DMA_CPARx if DIR = 1, else DMA_CMARx)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3</td>
<td>1: read B0[7:0] @0x0 then write B0[7:0] @0x0 2: read B1[7:0] @0x1 then write B1[7:0] @0x1 3: read B2[7:0] @0x2 then write B2[7:0] @0x2 4: read B3[7:0] @0x3 then write B3[7:0] @0x3</td>
<td>@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>4</td>
<td>@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3</td>
<td>1: read B0[7:0] @0x0 then write 00B0[15:0] @0x0 2: read B1[7:0] @0x1 then write 00B1[15:0] @0x2 3: read B2[7:0] @0x2 then write 00B2[15:0] @0x4 4: read B3[7:0] @0x3 then write 00B3[15:0] @0x6</td>
<td>@0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>4</td>
<td>@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3</td>
<td>1: read B0[7:0] @0x0 then write 000000B0[31:0] @0x0 2: read B1[7:0] @0x1 then write 000000B1[31:0] @0x4 3: read B2[7:0] @0x2 then write 000000B2[31:0] @0x8 4: read B3[7:0] @0x3 then write 000000B3[31:0] @0xC</td>
<td>@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>4</td>
<td>@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6</td>
<td>1: read B1B0[15:0] @0x0 then write B0[7:0] @0x0 2: read B3B2[15:0] @0x2 then write B2[7:0] @0x1 3: read B5B4[15:0] @0x4 then write B4[7:0] @0x2 4: read B7B6[15:0] @0x6 then write B6[7:0] @0x3</td>
<td>@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x6 / B6</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>4</td>
<td>@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6</td>
<td>1: read B1B0[15:0] @0x0 then write B1B0[15:0] @0x0 2: read B3B2[15:0] @0x2 then write B3B2[15:0] @0x2 3: read B5B4[15:0] @0x4 then write B5B4[15:0] @0x4 4: read B7B6[15:0] @0x6 then write B7B6[15:0] @0x6</td>
<td>@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6</td>
</tr>
<tr>
<td>16</td>
<td>32</td>
<td>4</td>
<td>@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6</td>
<td>1: read B1B0[15:0] @0x0 then write 0000B1B0[31:0] @0x0 2: read B3B2[15:0] @0x2 then write 0000B3B2[31:0] @0x4 3: read B5B4[15:0] @0x4 then write 0000B5B4[31:0] @0x8 4: read B7B6[15:0] @0x6 then write 0000B7B6[31:0] @0xC</td>
<td>@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6</td>
</tr>
<tr>
<td>32</td>
<td>8</td>
<td>4</td>
<td>@0x0 / B3B2B1B0 @0x2 / B7B6B5B4 @0x4 / BBAB9B9B8 @0x6 / BFBEBD8C</td>
<td>1: read B3B2B1B0[31:0] @0x0 then write B0[7:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B4[7:0] @0x1 3: read BBAB9B9B8[31:0] @0x8 then write B8[7:0] @0x2 4: read BFBEBD8C[31:0] @0xC then write BC[7:0] @0x3</td>
<td>@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC</td>
</tr>
<tr>
<td>32</td>
<td>16</td>
<td>4</td>
<td>@0x0 / B3B2B1B0 @0x2 / B7B6B5B4 @0x4 / BBAB9B9B8 @0x6 / BFBEBD8C</td>
<td>1: read B3B2B1B0[31:0] @0x0 then write B1B0[15:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B5B4[15:0] @0x2 3: read BBAB9B9B8[31:0] @0x8 then write BBAB9B9B8[31:0] @0x8 4: read BFBEBD8C[31:0] @0xC then write BFBEBD8C[31:0] @0xC</td>
<td>@0x0 / B1B0 @0x2 / B5B4 @0x4 / BBAB9B9B8 @0xC / BFBEBD8C</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>4</td>
<td>@0x0 / B3B2B1B0 @0x2 / B7B6B5B4 @0x4 / BBAB9B9B8 @0x6 / BFBEBD8C</td>
<td>1: read B3B2B1B0[31:0] @0x0 then write B3B2B1B0[31:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B7B6B5B4[31:0] @0x4 3: read BBAB9B9B8[31:0] @0x8 then write BBAB9B9B8[31:0] @0x8 4: read BFBEBD8C[31:0] @0xC then write BFBEBD8C[31:0] @0xC</td>
<td>@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBAB9B9B8 @0xC / BFBEBD8C</td>
</tr>
</tbody>
</table>
Addressing AHB peripherals not supporting byte/half-word write transfers

When the DMA controller initiates an AHB byte or half-word write transfer, the data are duplicated on the unused lanes of the AHB master 32-bit data bus (HWDATA[31:0]).

When the AHB slave peripheral does not support byte or half-word write transfers and does not generate any error, the DMA controller writes the 32 HWDATA bits as shown in the two examples below:

- To write the half-word 0xABCD, the DMA controller sets the HWDATA bus to 0xABCDABCD with a half-word data size (HSIZE = HalfWord in AHB master bus).
- To write the byte 0xAB, the DMA controller sets the HWDATA bus to 0xABABABAB with a byte data size (HSIZE = Byte in the AHB master bus).

Assuming the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take into account the HSIZE data, any AHB byte or half-word transfer is changed into a 32-bit APB transfer as described below:

- An AHB byte write transfer of 0xB0 to one of the 0x0, 0x1, 0x2 or 0x3 addresses, is converted to an APB word write transfer of 0xB0B0B0B0 to the 0x0 address.
- An AHB half-word write transfer of 0xB1B0 to the 0x0 or 0x2 addresses, is converted to an APB word write transfer of 0xB1B0B1B0 to the 0x0 address.

9.4.7 DMA error management

A DMA transfer error is generated when reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or write access, the faulty channel x is automatically disabled through a hardware clear of its EN bit in the corresponding DMA_CCRx register.

The TEIFx bit of the DMA_ISR register is set. An interrupt is then generated if the TEIE bit of the DMA_CCRx register is set.

The EN bit of the DMA_CCRx register can not be set again by software (channel x re-activated) until the TEIFx bit of the DMA_ISR register is cleared (by setting the CTEIFx bit of the DMA_IFCR register).

When the software is notified with a transfer error over a channel which involves a peripheral, the software has first to stop this peripheral in DMA mode, in order to disable any pending or future DMA request. Then software may normally reconfigure both DMA and the peripheral in DMA mode for a new transfer.
9.5 DMA interrupts

An interrupt can be generated on a half transfer, transfer complete or transfer error for each DMA channel x. Separate interrupt enable bits are available for flexibility.

Table 39. DMA interrupt requests

<table>
<thead>
<tr>
<th>Interrupt request</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Interrupt enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel x interrupt</td>
<td>Half transfer on channel x</td>
<td>HTIFx</td>
<td>HTIEEx</td>
</tr>
<tr>
<td></td>
<td>Transfer complete on channel x</td>
<td>TCIFx</td>
<td>TCIEx</td>
</tr>
<tr>
<td></td>
<td>Transfer error on channel x</td>
<td>TEIFx</td>
<td>TEIEEx</td>
</tr>
<tr>
<td></td>
<td>Half transfer or transfer complete or transfer error on channel x</td>
<td>GIFx</td>
<td>-</td>
</tr>
</tbody>
</table>

9.6 DMA registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

The DMA registers have to be accessed by words (32-bit).

9.6.1 DMA interrupt status register (DMA_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

Every status bit is cleared by hardware when the software sets the corresponding clear bit or the corresponding global clear bit CGIFx, in the DMA_IFCR register.

```
    31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
     r r r  r  r  r  r  r  r  r  r  r  r  r  r  r
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
    TEIF4 HTIF4 TCIF4 GIF4 TEIF3 HTIF3 TCIF3 GIF3 TEIF2 HTIF2 TCIF2 GIF2 TEIF1 HTIF1 TCIF1 GIF1
     r r r  r  r  r  r  r  r  r  r  r  r  r  r  r
```

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

- Bit 27 **TEIF7**: transfer error (TE) flag for channel 7
  0: no TE event
  1: a TE event occurred

- Bit 26 **HTIF7**: half transfer (HT) flag for channel 7
  0: no HT event
  1: a HT event occurred

- Bit 25 **TCIF7**: transfer complete (TC) flag for channel 7
  0: no TC event
  1: a TC event occurred

- Bit 24 **GIF7**: global interrupt flag for channel 7
  0: no TE, HT or TC event
  1: a TE, HT or TC event occurred
Bit 23  **TEIF6**: transfer error (TE) flag for channel 6  
0: no TE event  
1: a TE event occurred  

Bit 22  **HTIF6**: half transfer (HT) flag for channel 6  
0: no HT event   
1: a HT event occurred  

Bit 21  **TCIF6**: transfer complete (TC) flag for channel 6  
0: no TC event   
1: a TC event occurred  

Bit 20  **GIF6**: global interrupt flag for channel 6  
0: no TE, HT or TC event  
1: a TE, HT or TC event occurred  

Bit 19  **TEIF5**: transfer error (TE) flag for channel 5  
0: no TE event   
1: a TE event occurred  

Bit 18  **HTIF5**: half transfer (HT) flag for channel 5  
0: no HT event   
1: a HT event occurred  

Bit 17  **TCIF5**: transfer complete (TC) flag for channel 5  
0: no TC event   
1: a TC event occurred  

Bit 16  **GIF5**: global interrupt flag for channel 5  
0: no TE, HT or TC event   
1: a TE, HT or TC event occurred  

Bit 15  **TEIF4**: transfer error (TE) flag for channel 4  
0: no TE event   
1: a TE event occurred  

Bit 14  **HTIF4**: half transfer (HT) flag for channel 4  
0: no HT event   
1: a HT event occurred  

Bit 13  **TCIF4**: transfer complete (TC) flag for channel 4  
0: no TC event   
1: a TC event occurred  

Bit 12  **GIF4**: global interrupt flag for channel 4  
0: no TE, HT or TC event   
1: a TE, HT or TC event occurred  

Bit 11  **TEIF3**: transfer error (TE) flag for channel 3  
0: no TE event   
1: a TE event occurred  

Bit 10  **HTIF3**: half transfer (HT) flag for channel 3  
0: no HT event   
1: a HT event occurred  

Bit 9   **TCIF3**: transfer complete (TC) flag for channel 3  
0: no TC event   
1: a TC event occurred
Bit 8 **GIF3**: global interrupt flag for channel 3
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 7 **TEIF2**: transfer error (TE) flag for channel 2
0: no TE event
1: a TE event occurred

Bit 6 **HTIF2**: half transfer (HT) flag for channel 2
0: no HT event
1: a HT event occurred

Bit 5 **TCIF2**: transfer complete (TC) flag for channel 2
0: no TC event
1: a TC event occurred

Bit 4 **GIF2**: global interrupt flag for channel 2
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 3 **TEIF1**: transfer error (TE) flag for channel 1
0: no TE event
1: a TE event occurred

Bit 2 **HTIF1**: half transfer (HT) flag for channel 1
0: no HT event
1: a HT event occurred

Bit 1 **TCIF1**: transfer complete (TC) flag for channel 1
0: no TC event
1: a TC event occurred

Bit 0 **GIF1**: global interrupt flag for channel 1
0: no TE, HT or TC event
1: a TE, HT or TC event occurred
9.6.2 DMA interrupt flag clear register (DMA_IFCR)

Address offset: 0x04
Reset value: 0x0000 0000

Setting the global clear bit CGIFx of the channel x in this DMA_IFCR register, causes the DMA hardware to clear the corresponding GIFx bit and any individual flag among TEIFx, HTIFx, TCIFx, in the DMA_ISR register.

Setting any individual clear bit among CTEIFx, CHTIFx, CTCIFx in this DMA_IFCR register, causes the DMA hardware to clear the corresponding individual flag and the global flag GIFx in the DMA_ISR register, provided that none of the two other individual flags is set.

Writing 0 into any flag clear bit has no effect.

<table>
<thead>
<tr>
<th>Res.</th>
<th>Res.</th>
<th>Res.</th>
<th>Res.</th>
<th>CTEIF7</th>
<th>CHTIF7</th>
<th>CTCIF7</th>
<th>CGIF7</th>
<th>CTEIF6</th>
<th>CHTIF6</th>
<th>CTCIF6</th>
<th>CGIF6</th>
<th>CTEIF5</th>
<th>CHTIF5</th>
<th>CTCIF5</th>
<th>CGIF5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>

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

- Bit 27 CTEIF7: transfer error flag clear for channel 7
- Bit 26 CHTIF7: half transfer flag clear for channel 7
- Bit 25 CTCIF7: transfer complete flag clear for channel 7
- Bit 24 CGIF7: global interrupt flag clear for channel 7
- Bit 23 CTEIF6: transfer error flag clear for channel 6
- Bit 22 CHTIF6: half transfer flag clear for channel 6
- Bit 21 CTCIF6: transfer complete flag clear for channel 6
- Bit 20 CGIF6: global interrupt flag clear for channel 6
- Bit 19 CTEIF5: transfer error flag clear for channel 5
- Bit 18 CHTIF5: half transfer flag clear for channel 5
- Bit 17 CTCIF5: transfer complete flag clear for channel 5
- Bit 16 CGIF5: global interrupt flag clear for channel 5
- Bit 15 CTEIF4: transfer error flag clear for channel 4
- Bit 14 CHTIF4: half transfer flag clear for channel 4
- Bit 13 CTCIF4: transfer complete flag clear for channel 4
- Bit 12 CGIF4: global interrupt flag clear for channel 4
- Bit 11 CTEIF3: transfer error flag clear for channel 3
- Bit 10 CHTIF3: half transfer flag clear for channel 3
- Bit 9 CTCIF3: transfer complete flag clear for channel 3
Bit 8 **CGIF3**: global interrupt flag clear for channel 3  
Bit 7 **CTEIF2**: transfer error flag clear for channel 2  
Bit 6 **CHTIF2**: half transfer flag clear for channel 2  
Bit 5 **CTCIF2**: transfer complete flag clear for channel 2  
Bit 4 **CGIF2**: global interrupt flag clear for channel 2  
Bit 3 **CTEIF1**: transfer error flag clear for channel 1  
Bit 2 **CHTIF1**: half transfer flag clear for channel 1  
Bit 1 **CTCIF1**: transfer complete flag clear for channel 1  
Bit 0 **CGIF1**: global interrupt flag clear for channel 1

### 9.6.3 DMA channel x configuration register (DMA_CCRx)

Address offset: 0x08 + 0x14 * (x - 1), (x = 1 to 7)  
Reset value: 0x0000 0000

The register fields/bits MEM2MEM, PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, and DIR are read-only when EN = 1.

The states of MEM2MEM and CIRC bits must not be both high at the same time.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM2MEM</td>
<td>PL[1:0]</td>
<td>MSIZE[1:0]</td>
<td>PSIZE[1:0]</td>
<td>MINC</td>
<td>PINC</td>
<td>CIRC</td>
<td>DIR</td>
<td>TEIE</td>
<td>HTIE</td>
<td>TCIE</td>
<td>EN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **MEM2MEM**: memory-to-memory mode  
0: disabled  
1: enabled  

*Note*: this bit is set and cleared by software.  
*It must not be written when the channel is enabled (EN = 1).*  
*It is read-only when the channel is enabled (EN = 1).*

Bits 13:12 **PL[1:0]**: priority level  
00: low  
01: medium  
10: high  
11: very high  

*Note*: this field is set and cleared by software.  
*It must not be written when the channel is enabled (EN = 1).*  
*It is read-only when the channel is enabled (EN = 1).*
Bits 11:10  **MSIZE[1:0]:** memory size

Defines the data size of each DMA transfer to the identified memory.

In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the memory destination if DIR = 0.
In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.
00: 8 bits
01: 16 bits
10: 32 bits
11: reserved

*Note:* this field is set and cleared by software.
*It must not be written when the channel is enabled (EN = 1).*
*It is read-only when the channel is enabled (EN = 1).*

Bits 9:8  **PSIZE[1:0]:** peripheral size

Defines the data size of each DMA transfer to the identified peripheral.

In memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.
In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.
00: 8 bits
01: 16 bits
10: 32 bits
11: reserved

*Note:* this field is set and cleared by software.
*It must not be written when the channel is enabled (EN = 1).*
*It is read-only when the channel is enabled (EN = 1).*

Bit 7  **MINC:** memory increment mode

Defines the increment mode for each DMA transfer to the identified memory.

In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the memory destination if DIR = 0.
In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.
0: disabled
1: enabled

*Note:* this bit is set and cleared by software.
*It must not be written when the channel is enabled (EN = 1).*
*It is read-only when the channel is enabled (EN = 1).*

Bit 6  **PINC:** peripheral increment mode

Defines the increment mode for each DMA transfer to the identified peripheral.

In memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.
In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.
0: disabled
1: enabled

*Note:* this bit is set and cleared by software.
*It must not be written when the channel is enabled (EN = 1).*
*It is read-only when the channel is enabled (EN = 1).*
Bit 5 **CIRC**: circular mode  
0: disabled  
1: enabled  
  *Note*: this bit is set and cleared by software.  
  *It must not be written when the channel is enabled (EN = 1).*  
  *It is not read-only when the channel is enabled (EN = 1).*

Bit 4 **DIR**: data transfer direction  
This bit must be set only in memory-to-peripheral and peripheral-to-memory modes.  
0: read from peripheral  
  – Source attributes are defined by PSIZE and PINC, plus the DMA_CPARx register.  
    This is still valid in a memory-to-memory mode.  
  – Destination attributes are defined by MSIZE and MINC, plus the DMA_CMARx register. This is still valid in a peripheral-to-peripheral mode.  
1: read from memory  
  – Destination attributes are defined by PSIZE and PINC, plus the DMA_CPARx register. This is still valid in a memory-to-memory mode.  
  – Source attributes are defined by MSIZE and MINC, plus the DMA_CMARx register. This is still valid in a peripheral-to-peripheral mode.  
  *Note*: this bit is set and cleared by software.  
  *It must not be written when the channel is enabled (EN = 1).*  
  *It is read-only when the channel is enabled (EN = 1).*

Bit 3 **TEIE**: transfer error interrupt enable  
0: disabled  
1: enabled  
  *Note*: this bit is set and cleared by software.  
  *It must not be written when the channel is enabled (EN = 1).*  
  *It is not read-only when the channel is enabled (EN = 1).*

Bit 2 **HTIE**: half transfer interrupt enable  
0: disabled  
1: enabled  
  *Note*: this bit is set and cleared by software.  
  *It must not be written when the channel is enabled (EN = 1).*  
  *It is not read-only when the channel is enabled (EN = 1).*

Bit 1 **TCIE**: transfer complete interrupt enable  
0: disabled  
1: enabled  
  *Note*: this bit is set and cleared by software.  
  *It must not be written when the channel is enabled (EN = 1).*  
  *It is not read-only when the channel is enabled (EN = 1).*

Bit 0 **EN**: channel enable  
When a channel transfer error occurs, this bit is cleared by hardware. It can not be set again by software (channel x re-activated) until the TEIFx bit of the DMA_ISR register is cleared (by setting the CTEIFx bit of the DMA_IFCR register).  
0: disabled  
1: enabled  
  *Note*: this bit is set and cleared by software.
9.6.4 DMA channel x number of data to transfer register (DMA_CNDTRx)

Address offset: 0x0C + 0x14 * (x - 1), (x = 1 to 7)
Reset value: 0x0000 0000

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **NDT[15:0]**: number of data to transfer (0 to 216 - 1)
This field is updated by hardware when the channel is enabled:
- It is decremented after each single DMA ‘read followed by write’ transfer, indicating the remaining amount of data items to transfer.
- It is kept at zero when the programmed amount of data to transfer is reached, if the channel is not in circular mode (CIRC = 0 in the DMA_CCRx register).
- It is reloaded automatically by the previously programmed value, when the transfer is complete, if the channel is in circular mode (CIRC = 1).

If this field is zero, no transfer can be served whatever the channel status (enabled or not).

**Note:** this field is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

9.6.5 DMA channel x peripheral address register (DMA_CPARx)

Address offset: 0x10 + 0x14 * (x - 1), (x = 1 to 7)
Reset value: 0x0000 0000
Bits 31:0  **PA[31:0]:** peripheral address  
It contains the base address of the peripheral data register from/to which the data will be read/written.  
When PSIZE[1:0] = 01 (16 bits), bit 0 of PA[31:0] is ignored. Access is automatically aligned to a half-word address.  
When PSIZE = 10 (32 bits), bits 1 and 0 of PA[31:0] are ignored. Access is automatically aligned to a word address.  
In memory-to-memory mode, this register identifies the memory destination address if DIR = 1 and the memory source address if DIR = 0.  
In peripheral-to-peripheral mode, this register identifies the peripheral destination address DIR = 1 and the peripheral source address if DIR = 0.  
*Note:* this register is set and cleared by software.  
*It must not be written when the channel is enabled (EN = 1).*  
*It is not read-only when the channel is enabled (EN = 1).*

### 9.6.6 DMA channel x memory address register (DMA_CMARx)

Address offset: 0x14 + 0x14 \* (x - 1), (x = 1 to 7)  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>R31 R30 R29 R28 R27 R26 R25 R24 R23 R22 R21 R20 R19 R18 R17 R16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>DMA_ISR</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

### 9.6.7 DMA register map and reset values

*Table 40* gives the DMA register map and reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>R31 R30 R29 R28 R27 R26 R25 R24 R23 R22 R21 R20 R19 R18 R17 R16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>DMA_ISR</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Reset value: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
### Table 40. DMA register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x004</td>
<td>DMA_IFCR</td>
<td>0x008</td>
<td>DMA_CCR1</td>
<td>0x00C</td>
<td>DMA_CNDTR1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x008</td>
<td>DMA_CCR2</td>
<td>0x010</td>
<td>DMA_CPAR1</td>
<td>0x014</td>
<td>DMA_CMAR1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x010</td>
<td>DMA_CNDTR2</td>
<td>0x014</td>
<td>DMA_CMAR2</td>
<td>0x018</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x014</td>
<td>DMA_CCR3</td>
<td>0x020</td>
<td>DMA_CNDTR3</td>
<td>0x024</td>
<td>DMA_CPAR2</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x020</td>
<td>DMA_CNDTR3</td>
<td>0x028</td>
<td>DMA_CMAR2</td>
<td>0x02C</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x028</td>
<td>DMA_CMAR2</td>
<td>0x030</td>
<td>DMA_CCR3</td>
<td>0x034</td>
<td>DMA_CNDTR3</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x030</td>
<td>DMA_CCR3</td>
<td>0x038</td>
<td>DMA_CPAR3</td>
<td>0x03C</td>
<td>DMA_CMAR3</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x038</td>
<td>DMA_CPAR3</td>
<td>0x040</td>
<td>Reserved</td>
<td>0x044</td>
<td>DMA_CCR4</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x040</td>
<td>Reserved</td>
<td>0x044</td>
<td>DMA_CCR4</td>
<td>0x048</td>
<td>DMA_CNDTR4</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x044</td>
<td>DMA_CCR4</td>
<td>0x048</td>
<td>DMA_CNDTR4</td>
<td>0x04C</td>
<td>DMA_CPAR4</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x048</td>
<td>DMA_CNDTR4</td>
<td>0x04C</td>
<td>DMA_CPAR4</td>
<td>0x050</td>
<td>DMA_CMAR4</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x050</td>
<td>DMA_CMAR4</td>
<td>0x054</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Note:** The table continues with more registers and reset values.
Table 40. DMA register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Reset value</th>
<th>MEM/MEM</th>
<th>PL[1:0]</th>
<th>MSIZE[1:0]</th>
<th>PSIZE[1:0]</th>
<th>PINC</th>
<th>PNIE</th>
<th>DIR</th>
<th>CIRC</th>
<th>TEIE</th>
<th>HTIE</th>
<th>TCIE</th>
<th>EN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x058</td>
<td>DMA_CCR5</td>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x05C</td>
<td>DMA_CNDTR5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x060</td>
<td>DMA_CPAR5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x064</td>
<td>DMA_CMAR5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x068</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x06C</td>
<td>DMA_CCR6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x070</td>
<td>DMA_CNDTR6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x074</td>
<td>DMA_CPAR6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x078</td>
<td>DMA_CMAR6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x07C</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x080</td>
<td>DMA_CCR7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x084</td>
<td>DMA_CNDTR7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x088</td>
<td>DMA_CPAR7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08C</td>
<td>DMA_CMAR7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.2 for the register boundary addresses.
10 DMA request multiplexer (DMAMUX)

10.1 Introduction

A peripheral indicates a request for DMA transfer by setting its DMA request signal. The DMA request is pending until it is served by the DMA controller that generates a DMA acknowledge signal, and the corresponding DMA request signal is de-asserted.

In this document, the set of control signals required for the DMA request/acknowledge protocol is not explicitly shown or described, and it is referred to as DMA request line.

The DMAMUX request multiplexer enables routing a DMA request line between the peripherals and the DMA controllers of the product. The routing function is ensured by a programmable multi-channel DMA request line multiplexer. Each channel selects a unique DMA request line, unconditionally or synchronously with events from its DMAMUX synchronization inputs. The DMAMUX may also be used as a DMA request generator from programmable events on its input trigger signals.

The number of DMAMUX instances and their main characteristics are specified in Section 10.3.1.

The assignment of DMAMUX request multiplexer inputs to the DMA request lines from peripherals and to the DMAMUX request generator outputs, the assignment of DMAMUX request multiplexer outputs to DMA controller channels, and the assignment of DMAMUX synchronizations and trigger inputs to internal and external signals depend on the product implementation, and are detailed in Section 10.3.2.
10.2 DMAMUX main features

- 7-channel programmable DMA request line multiplexer output
- 4-channel DMA request generator
- 23 trigger inputs to DMA request generator
- 23 synchronization inputs
- Per DMA request generator channel:
  - DMA request trigger input selector
  - DMA request counter
  - Event overrun flag for selected DMA request trigger input
- Per DMA request line multiplexer channel output:
  - 57 input DMA request lines from peripherals
  - One DMA request line output
  - Synchronization input selector
  - DMA request counter
  - Event overrun flag for selected synchronization input
  - One event output, for DMA request chaining

10.3 DMAMUX implementation

10.3.1 DMAMUX instantiation

DMAMUX is instantiated with the hardware configuration parameters listed in the following table.

<table>
<thead>
<tr>
<th>Feature</th>
<th>DMAMUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of DMAMUX output request channels</td>
<td>7/5(1)</td>
</tr>
<tr>
<td>Number of DMAMUX request generator channels</td>
<td>4</td>
</tr>
<tr>
<td>Number of DMAMUX request trigger inputs</td>
<td>23</td>
</tr>
<tr>
<td>Number of DMAMUX synchronization inputs</td>
<td>23</td>
</tr>
<tr>
<td>Number of DMAMUX peripheral request inputs</td>
<td>57</td>
</tr>
</tbody>
</table>

1. Seven channels for STM32G071xx and STM32G081xx, five for STM32G031xx and STM32G041xx.

10.3.2 DMAMUX mapping

The mapping of resources to DMAMUX is hardwired.
### Table 42. DMAMUX: assignment of multiplexer inputs to resources

<table>
<thead>
<tr>
<th>DMA request MUX input</th>
<th>Resource</th>
<th>DMA request MUX input</th>
<th>Resource</th>
<th>DMA request MUX input</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dmamux_req_gen0</td>
<td>22</td>
<td>TIM1_CH3</td>
<td>43</td>
<td>TIM15_UP</td>
</tr>
<tr>
<td>2</td>
<td>dmamux_req_gen1</td>
<td>23</td>
<td>TIM1_CH4</td>
<td>44</td>
<td>TIM16_CH1</td>
</tr>
<tr>
<td>3</td>
<td>dmamux_req_gen2</td>
<td>24</td>
<td>TIM1_TRIG_COM</td>
<td>45</td>
<td>TIM16_TRIG_COM</td>
</tr>
<tr>
<td>4</td>
<td>dmamux_req_gen3</td>
<td>25</td>
<td>TIM1_UP</td>
<td>46</td>
<td>TIM16_UP</td>
</tr>
<tr>
<td>5</td>
<td>ADC</td>
<td>26</td>
<td>TIM2_CH1</td>
<td>47</td>
<td>TIM17_CH1</td>
</tr>
<tr>
<td>6</td>
<td>AES_IN</td>
<td>27</td>
<td>TIM2_CH2</td>
<td>48</td>
<td>TIM17_TRIG_COM</td>
</tr>
<tr>
<td>7</td>
<td>AES_OUT</td>
<td>28</td>
<td>TIM2_CH3</td>
<td>49</td>
<td>TIM17_UP</td>
</tr>
<tr>
<td>8</td>
<td>DAC_Channel1</td>
<td>29</td>
<td>TIM2_CH4</td>
<td>50</td>
<td>USART1_RX</td>
</tr>
<tr>
<td>9</td>
<td>DAC_Channel2</td>
<td>30</td>
<td>TIM2_TRIG</td>
<td>51</td>
<td>USART1_TX</td>
</tr>
<tr>
<td>10</td>
<td>I2C1_RX</td>
<td>31</td>
<td>TIM2_UP</td>
<td>52</td>
<td>USART2_RX</td>
</tr>
<tr>
<td>11</td>
<td>I2C1_TX</td>
<td>32</td>
<td>TIM3_CH1</td>
<td>53</td>
<td>USART2_TX</td>
</tr>
<tr>
<td>12</td>
<td>I2C2_RX</td>
<td>33</td>
<td>TIM3_CH2</td>
<td>54</td>
<td>USART3_RX</td>
</tr>
<tr>
<td>13</td>
<td>I2C2_TX</td>
<td>34</td>
<td>TIM3_CH3</td>
<td>55</td>
<td>USART3_TX</td>
</tr>
<tr>
<td>14</td>
<td>LPUART_RX</td>
<td>35</td>
<td>TIM3_CH4</td>
<td>56</td>
<td>USART4_RX</td>
</tr>
<tr>
<td>15</td>
<td>LPUART_TX</td>
<td>36</td>
<td>TIM3_TRIG</td>
<td>57</td>
<td>USART4_TX</td>
</tr>
<tr>
<td>16</td>
<td>SPI1_RX</td>
<td>37</td>
<td>TIM3_UP</td>
<td>58</td>
<td>UCPD1_RX</td>
</tr>
<tr>
<td>17</td>
<td>SPI1_TX</td>
<td>38</td>
<td>TIM6_UP</td>
<td>59</td>
<td>UCPD1_TX</td>
</tr>
<tr>
<td>18</td>
<td>SPI2_RX</td>
<td>39</td>
<td>TIM7_UP</td>
<td>60</td>
<td>UCPD2_RX</td>
</tr>
<tr>
<td>19</td>
<td>SPI2_TX</td>
<td>40</td>
<td>TIM15_CH1</td>
<td>61</td>
<td>UCPD2_TX</td>
</tr>
<tr>
<td>20</td>
<td>TIM1_CH1</td>
<td>41</td>
<td>TIM15_CH2</td>
<td>62</td>
<td>Reserved</td>
</tr>
<tr>
<td>21</td>
<td>TIM1_CH2</td>
<td>42</td>
<td>TIM15_TRIG_COM</td>
<td>63</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### Table 43. DMAMUX: assignment of trigger inputs to resources

<table>
<thead>
<tr>
<th>Trigger input</th>
<th>Resource</th>
<th>Trigger input</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EXTI LINE0</td>
<td>12</td>
<td>EXTI LINE12</td>
</tr>
<tr>
<td>1</td>
<td>EXTI LINE1</td>
<td>13</td>
<td>EXTI LINE13</td>
</tr>
<tr>
<td>2</td>
<td>EXTI LINE2</td>
<td>14</td>
<td>EXTI LINE14</td>
</tr>
<tr>
<td>3</td>
<td>EXTI LINE3</td>
<td>15</td>
<td>EXTI LINE15</td>
</tr>
<tr>
<td>4</td>
<td>EXTI LINE4</td>
<td>16</td>
<td>dmamux_evt0</td>
</tr>
<tr>
<td>5</td>
<td>EXTI LINE5</td>
<td>17</td>
<td>dmamux_evt1</td>
</tr>
<tr>
<td>6</td>
<td>EXTI LINE6</td>
<td>18</td>
<td>dmamux_evt2</td>
</tr>
<tr>
<td>7</td>
<td>EXTI LINE7</td>
<td>19</td>
<td>dmamux_evt3</td>
</tr>
<tr>
<td>8</td>
<td>EXTI LINE8</td>
<td>20</td>
<td>LPTIM1_OUT</td>
</tr>
<tr>
<td>9</td>
<td>EXTI LINE9</td>
<td>21</td>
<td>LPTIM2_OUT</td>
</tr>
</tbody>
</table>
Table 43. DMAMUX: assignment of trigger inputs to resources (continued)

<table>
<thead>
<tr>
<th>Trigger input</th>
<th>Resource</th>
<th>Trigger input</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>EXTI LINE10</td>
<td>22</td>
<td>TIM14_OC</td>
</tr>
<tr>
<td>11</td>
<td>EXTI LINE11</td>
<td>23</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 44. DMAMUX: assignment of synchronization inputs to resources

<table>
<thead>
<tr>
<th>Sync. input</th>
<th>Resource</th>
<th>Sync. input</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EXTI LINE0</td>
<td>12</td>
<td>EXTI LINE12</td>
</tr>
<tr>
<td>1</td>
<td>EXTI LINE1</td>
<td>13</td>
<td>EXTI LINE13</td>
</tr>
<tr>
<td>2</td>
<td>EXTI LINE2</td>
<td>14</td>
<td>EXTI LINE14</td>
</tr>
<tr>
<td>3</td>
<td>EXTI LINE3</td>
<td>15</td>
<td>EXTI LINE15</td>
</tr>
<tr>
<td>4</td>
<td>EXTI LINE4</td>
<td>16</td>
<td>dmamux_evt0</td>
</tr>
<tr>
<td>5</td>
<td>EXTI LINE5</td>
<td>17</td>
<td>dmamux_evt1</td>
</tr>
<tr>
<td>6</td>
<td>EXTI LINE6</td>
<td>18</td>
<td>dmamux_evt2</td>
</tr>
<tr>
<td>7</td>
<td>EXTI LINE7</td>
<td>19</td>
<td>dmamux_evt3</td>
</tr>
<tr>
<td>8</td>
<td>EXTI LINE8</td>
<td>20</td>
<td>LPTIM1_OUT</td>
</tr>
<tr>
<td>9</td>
<td>EXTI LINE9</td>
<td>21</td>
<td>LPTIM2_OUT</td>
</tr>
<tr>
<td>10</td>
<td>EXTI LINE10</td>
<td>22</td>
<td>TIM14_OC</td>
</tr>
<tr>
<td>11</td>
<td>EXTI LINE11</td>
<td>23</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
10.4 DMAMUX functional description

10.4.1 DMAMUX block diagram

*Figure 21* shows the DMAMUX block diagram.

*Figure 21. DMAMUX block diagram*

DMAMUX features two main sub-blocks: the request line multiplexer and the request line generator.

The implementation assigns:
- DMAMUX request multiplexer sub-block inputs (dmamux_reqx) from peripherals (dmamux_req_inx) and from channels of the DMAMUX request generator sub-block (dmamux_req_genx)
- DMAMUX request outputs to channels of DMA controllers (dmamux_req_outx)
- Internal or external signals to DMA request trigger inputs (dmamux_trgx)
- Internal or external signals to synchronization inputs (dmamux_syncx)
10.4.2 DMAMUX signals

*Table 45* lists the DMAMUX signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dmamux_hclk</td>
<td>DMAMUX AHB clock</td>
</tr>
<tr>
<td>dmamux_req_inx</td>
<td>DMAMUX DMA request line inputs from peripherals</td>
</tr>
<tr>
<td>dmamux_trgx</td>
<td>DMAMUX DMA request triggers inputs (to request generator sub-block)</td>
</tr>
<tr>
<td>dmamux_req_genx</td>
<td>DMAMUX request generator sub-block channels outputs</td>
</tr>
<tr>
<td>dmamux_reqx</td>
<td>DMAMUX request multiplexer sub-block inputs (from peripheral requests and request generator channels)</td>
</tr>
<tr>
<td>dmamux_syncx</td>
<td>DMAMUX synchronization inputs (to request multiplexer sub-block)</td>
</tr>
<tr>
<td>dmamux_req_outx</td>
<td>DMAMUX requests outputs (to DMA controllers)</td>
</tr>
<tr>
<td>dmamux_evtx</td>
<td>DMAMUX events outputs</td>
</tr>
<tr>
<td>dmamux_ovr_it</td>
<td>DMAMUX overrun interrupts</td>
</tr>
</tbody>
</table>

10.4.3 DMAMUX channels

A DMAMUX channel is a DMAMUX request multiplexer channel that may include, depending on the selected input of the request multiplexer, an additional DMAMUX request generator channel.

A DMAMUX request multiplexer channel is connected and dedicated to one single channel of DMA controller(s).

**Channel configuration procedure**

The following sequence should be followed to configure both a DMAMUX x channel and the related DMA channel y:

1. Set and configure completely the DMA channel y, except enabling the channel y.
2. Set and configure completely the related DMAMUX y channel.
3. Last, activate the DMA channel y by setting the EN bit in the DMA y channel register.

10.4.4 DMAMUX request line multiplexer

The DMAMUX request multiplexer with its multiple channels ensures the actual routing of DMA request/acknowledge control signals, named DMA request lines.

Each DMA request line is connected in parallel to all the channels of the DMAMUX request line multiplexer.

A DMA request is sourced either from the peripherals or from the DMAMUX request generator.

The DMAMUX request line multiplexer channel x selects the DMA request line number as configured by the DMAREQ_ID field in the DMAMUX_CxCR register.
Note: The null value in the field DMAREQ_ID corresponds to no DMA request line selected. It is not allowed to configure a same non-null DMAREQ_ID to two different channels of the DMAMUX request line multiplexer.

On top of the DMA request selection, the synchronization mode and/or the event generation may be configured and enabled, if required.

**Synchronization mode and channel event generation**

Each DMAMUX request line multiplexer channel x can be individually synchronized by setting the synchronization enable (SE) bit in the DMAMUX_CxCR register.

DMAMUX has multiple synchronization inputs. The synchronization inputs are connected in parallel to all the channels of the request multiplexer.

The synchronization input is selected via the SYNC_ID field in the DMAMUX_CxCR register of a given channel x.

When a channel is in this synchronization mode, the selected input DMA request line is propagated to the multiplexer channel output, once is detected a programmable rising/falling edge on the selected input synchronization signal, via the SPOL[1:0] field of the DMAMUX_CxCR register.

Additionally, there is a programmable DMA request counter, internally to the DMAMUX request multiplexer, which may be used for the channel request output generation and also possibly for an event generation. An event generation on the channel x output is enabled through the EGE bit (event generation enable) of the DMAMUX_CxCR register.

As shown in Figure 23, upon the detected edge of the synchronization input, the pending selected input DMA request line is connected to the DMAMUX multiplexer channel x output.

**Note:** If a synchronization event occurs while there is no pending selected input DMA request line, it will be discarded. The following asserted input request lines will not be connected to the DMAMUX multiplexer channel output until a synchronization event occurs again.

From this point on, each time the connected DMAMUX request is served by the DMA controller (a served request is de-asserted), the DMAMUX request counter is decremented. At its underrun, the DMA request counter is automatically loaded with the value in NBREQ field of the DMAMUX_CxCR register and the input DMA request line is disconnected from the multiplexer channel x output.

Thus, the number of DMA requests transferred to the multiplexer channel x output following a detected synchronization event, is equal to the value in NBREQ field, plus one.

**Note:** The NBREQ field value shall only be written by software when both synchronization enable bit SE and event generation enable EGE bit of the corresponding multiplexer channel x are disabled.
If EGE is enabled, the multiplexer channel generates a channel event, as a pulse of one AHB clock cycle, when its DMA request counter is automatically reloaded with the value of the programmed NBREQ field, as shown in Figure 22 and Figure 23.
**Note:** If EGE is enabled and NBREQ = 0, an event is generated after each served DMA request.

**Note:** A synchronization event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.

*Upon writing into DMAMUX_CxCR register, the synchronization events are masked during three AHB clock cycles.*

**Synchronization overrun and interrupt**

If a new synchronization event occurs before the request counter underrun (the internal request counter programmed via the NBREQ field of the DMAMUX_CxCR register), the synchronization overrun flag bit SOFx is set in the DMAMUX_CSR status register.

**Note:** The request multiplexer channel x synchronization shall be disabled (DMAMUX_CxCR.SE = 0) at the completion of the use of the related channel of the DMA controller. Else, upon a new detected synchronization event, there will be a synchronization overrun due to the absence of a DMA acknowledge (that is, no served request) received from the DMA controller.

The overrun flag SOFx is reset by setting the associated clear synchronization overrun flag bit CSOFx in the DMAMUX_CFR register.

Setting the synchronization overrun flag generates an interrupt if the synchronization overrun interrupt enable bit SOIE is set in the DMAMUX_CxCR register.

**10.4.5 DMAMUX request generator**

The DMAMUX request generator produces DMA requests following trigger events on its DMA request trigger inputs.

The DMAMUX request generator has multiple channels. DMA request trigger inputs are connected in parallel to all channels.

The outputs of DMAMUX request generator channels are inputs to the DMAMUX request line multiplexer.

Each DMAMUX request generator channel x has an enable bit GE (generator enable) in the corresponding DMAMUX_RGxCR register.

The DMA request trigger input for the DMAMUX request generator channel x is selected through the SIG_ID (trigger signal ID) field in the corresponding DMAMUX_RGxCR register.

Trigger events on a DMA request trigger input can be rising edge, falling edge or either edge. The active edge is selected through the GPOL (generator polarity) field in the corresponding DMAMUX_RGxCR register.

Upon the trigger event, the corresponding generator channel starts generating DMA requests on its output. Each time the DMAMUX generated request is served by the connected DMA controller (a served request is de-asserted), a built-in (inside the DMAMUX request generator) DMA request counter is decremented. At its underrun, the request generator channel stops generating DMA requests and the DMA request counter will be automatically reloaded to its programmed value upon the next trigger event.

Thus, the number of DMA requests generated after the trigger event is GNBREQ + 1.
Note: The GNBREQ field value shall only be written by software when the enable GE bit of the corresponding generator channel x is disabled.

A trigger event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.

Upon writing into DMAMUX_RGxCR register, the trigger events are masked during three AHB clock cycles.

**Trigger overrun and interrupt**

If a new DMA request trigger event occurs before the DMAMUX request generator counter underrun (the internal counter programmed via the GNBREQ field of the DMAMUX_RGxCR register), and if the request generator channel x was enabled via GE, then the request trigger event overrun flag bit OFx is asserted by the hardware in the status DMAMUX_RGSR register.

**Note:** The request generator channel x shall be disabled (DMAMUX_RGxCR.GE = 0) at the completion of the usage of the related channel of the DMA controller. Else, upon a new detected trigger event, there will be a trigger overrun due to the absence of an acknowledge (that is, no served request) received from the DMA.

The overrun flag OFx is reset by setting the associated clear overrun flag bit COFx in the DMAMUX_RGCFR register.

Setting the DMAMUX request trigger overrun flag generates an interrupt if the DMA request trigger event overrun interrupt enable bit OIE is set in the DMAMUX_RGxCR register.

### 10.5 DMAMUX interrupts

An interrupt can be generated upon:
- a synchronization event overrun in each DMA request line multiplexer channel
- a trigger event overrun in each DMA request generator channel

For each case, per-channel individual interrupt enable, status and clear flag register bits are available.

#### Table 46. DMAMUX interrupts

<table>
<thead>
<tr>
<th>Interrupt signal</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Clear bit</th>
<th>Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>dmamuxovr_it</td>
<td>Synchronization event overrun on channel x of the DMAMUX request line multiplexer</td>
<td>SOFx</td>
<td>CSOFx</td>
<td>SOIE</td>
</tr>
<tr>
<td></td>
<td>Trigger event overrun on channel x of the DMAMUX request generator</td>
<td>OFx</td>
<td>COFx</td>
<td>OIE</td>
</tr>
</tbody>
</table>
10.6 **DMAMUX registers**

Refer to the table containing register boundary addresses for the DMAMUX base address.

DMAMUX registers may be accessed per (8-bit) byte, (16-bit) half-word, or (32-bit) word. The address shall be aligned with the data size.

10.6.1 **DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)**

Address offset: 0x000 + 0x04 * x (x = 0 to 6)

Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
<td>Bits</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SYNC_ID[4:0]**: Synchronization identification

Selects the synchronization input (see Table 44: DMAMUX: assignment of synchronization inputs to resources).

Bits 23:19 **NBREQ[4:0]**: Number of DMA requests minus 1 to forward

Defines the number of DMA requests to forward to the DMA controller after a synchronization event, and/or the number of DMA requests before an output event is generated.

This field shall only be written when both SE and EGE bits are low.

Bits 18:17 **SPOL[1:0]**: Synchronization polarity

Defines the edge polarity of the selected synchronization input:

00: no event, i.e. no synchronization nor detection.
01: rising edge
10: falling edge
11: rising and falling edge

Bit 16 **SE**: Synchronization enable

0: synchronization disabled
1: synchronization enabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **EGE**: Event generation enable

0: event generation disabled
1: event generation enabled

Bit 8 **SOIE**: Synchronization overrun interrupt enable

0: interrupt disabled
1: interrupt enabled
10.6.2 DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR)

Address offset: 0x080
Reset value: 0x0000 0000

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **SOF[6:0]**: Synchronization overrun event flag
The flag is set when a synchronization event occurs on a DMA request line multiplexer channel x, while the DMA request counter value is lower than NBREQ.
The flag is cleared by writing 1 to the corresponding CSOFx bit in DMAMUX_CFR register.

10.6.3 DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR)

Address offset: 0x084
Reset value: 0x0000 0000

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **CSOF[6:0]**: Clear synchronization overrun event flag
Writing 1 in each bit clears the corresponding overrun flag SOFx in the DMAMUX_CSR register.
10.6.4 DMAMUX request generator channel x configuration register (DMAMUX_RGxCR)

Address offset: 0x100 + 0x04 * x (x = 0 to 3)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-24</th>
<th>Bit 23-19</th>
<th>Bit 18-17</th>
<th>Bit 16</th>
<th>Bit 15-9</th>
<th>Bit 8</th>
<th>Bit 7-5</th>
<th>Bit 4-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:19 **GNBREQ[4:0]:** Number of DMA requests to be generated (minus 1)
Defines the number of DMA requests to be generated after a trigger event. The actual number of generated DMA requests is GNBREQ +1.

*Note: This field shall only be written when GE bit is disabled.*

Bits 18:17 **GPOL[1:0]:** DMA request generator trigger polarity
Defines the edge polarity of the selected trigger input
00: no event. I.e. none trigger detection nor generation.
01: rising edge
10: falling edge
11: rising and falling edge

Bit 16 **GE:** DMA request generator channel x enable
0: DMA request generator channel x disabled
1: DMA request generator channel x enabled

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

Bit 8 **OIE:** Trigger overrun interrupt enable
0: interrupt on a trigger overrun event occurrence is disabled
1: interrupt on a trigger overrun event occurrence is enabled

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **SIG_ID[4:0]:** Signal identification
Selects the DMA request trigger input used for the channel x of the DMA request generator
### 10.6.5 DMAMUX request generator interrupt status register (DMAMUX_RGSR)

Address offset: 0x140  
Reset value: 0x0000 0000

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

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

Bits 3:0 **OF[3:0]**: Trigger overrun event flag

The flag is set when a new trigger event occurs on DMA request generator channel x, before the request counter underrun (the internal request counter programmed via the GNBREQ field of the DMAMUX_RGxC register).

The flag is cleared by writing 1 to the corresponding COFx bit in the DMAMUX_RGCFR register.

### 10.6.6 DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR)

Address offset: 0x144  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>OF3</td>
<td>OF2</td>
<td>OF1</td>
<td>OF0</td>
<td>COF3</td>
<td>COF2</td>
<td>COF1</td>
<td>COF0</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

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

Bits 3:0 **COF[3:0]**: Clear trigger overrun event flag

Writing 1 in each bit clears the corresponding overrun flag OFx in the DMAMUX_RGSR register.
### DMAMUX register map

The following table summarizes the DMAMUX registers and reset values. Refer to the register boundary address table for the DMAMUX register base address.

#### Table 47. DMAMUX register map and reset values

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | DMAMUX_C0CR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x04   | DMAMUX_C1CR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x08   | DMAMUX_C2CR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x0C   | DMAMUX_C3CR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x10   | DMAMUX_C4CR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x14   | DMAMUX_C5CR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x18   | DMAMUX_C6CR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x1C-0x1FC | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x080  | DMAMUX_CSR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x084  | DMAMUX_CFR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x088-0x0FC | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x100  | DMAMUX_RG0CR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x104  | DMAMUX_RG1CR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x108  | DMAMUX_RG2CR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x10C  | DMAMUX_RG3CR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x110-0x13C | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x140  | DMAMUX_RGSR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
Refer to *Section 2.2 on page 54* for the register boundary addresses.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x144</td>
<td>DMAMUX_GSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x148 - 0x3FC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 47. DMAMUX register map and reset values (continued)
11 Nested vectored interrupt controller (NVIC)

11.1 Main features

- 32 maskable interrupt channels (not including the sixteen Cortex®-M0+ interrupt lines)
- 4 programmable priority levels (2 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the processor core interface are closely coupled, which enables low-latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to the programming manual PM0223.

11.2 SysTick calibration value register

The SysTick calibration value is set to 6500, which gives a reference time base of 1 ms with the SysTick clock set to 6.5 MHz (max fHCLK/8).

11.3 Interrupt and exception vectors

*Table 48* is the vector table. Information pertaining to a peripheral only applies to devices containing that peripheral.

<table>
<thead>
<tr>
<th>Position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>Reserved</td>
<td>0x0000_0000</td>
</tr>
<tr>
<td>-</td>
<td>-3</td>
<td>fixed</td>
<td>Reset</td>
<td>Reset</td>
<td>0x0000_0004</td>
</tr>
<tr>
<td>-</td>
<td>-2</td>
<td>fixed</td>
<td>NMI_Handler</td>
<td>Non maskable interrupt. The SRAM parity err., Flash ECC double err., HSE CSS and LSE CSS are linked to the NMI vector.</td>
<td>0x0000_0008</td>
</tr>
<tr>
<td>-</td>
<td>-1</td>
<td>fixed</td>
<td>HardFault_Handler</td>
<td>All class of fault</td>
<td>0x0000_000C</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>Reserved</td>
<td>0x0000_0010 0x0000_0014 0x0000_0018 0x0000_001C 0x0000_0020 0x0000_0024 0x0000_0028</td>
</tr>
<tr>
<td>-</td>
<td>3</td>
<td>settable</td>
<td>SVC_Handler</td>
<td>System service call via SWI instruction</td>
<td>0x0000_002C</td>
</tr>
</tbody>
</table>
### Nested vectored interrupt controller (NVIC)

**Table 48. Vector table(1) (continued)**

<table>
<thead>
<tr>
<th>Position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td></td>
<td>0x0000_0030</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td></td>
<td>0x0000_0034</td>
</tr>
<tr>
<td>-</td>
<td>5</td>
<td>settable</td>
<td>PendSV_Handler</td>
<td>Pendable request for system service</td>
<td>0x0000_0038</td>
</tr>
<tr>
<td>-</td>
<td>6</td>
<td>settable</td>
<td>SysTick_Handler</td>
<td>System tick timer</td>
<td>0x0000_003C</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>settable</td>
<td>WWDG</td>
<td>Window watchdog interrupt</td>
<td>0x0000_0040</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>settable</td>
<td>PVD</td>
<td>Power voltage detector interrupt (EXTI line 16)</td>
<td>0x0000_0044</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>settable</td>
<td>RTC / TAMP</td>
<td>RTC and TAMP interrupts (combined EXTI lines 19 &amp; 21)</td>
<td>0x0000_0048</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>settable</td>
<td>FLASH</td>
<td>Flash global interrupt</td>
<td>0x0000_004C</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>settable</td>
<td>RCC</td>
<td>RCC global interrupt</td>
<td>0x0000_0050</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>settable</td>
<td>EXTI0_1</td>
<td>EXTI line 0 &amp; 1 interrupt</td>
<td>0x0000_0054</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>settable</td>
<td>EXTI2_3</td>
<td>EXTI line 2 &amp; 3 interrupt</td>
<td>0x0000_0058</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>settable</td>
<td>EXTI4_15</td>
<td>EXTI line 4 to 15 interrupt</td>
<td>0x0000_005C</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>settable</td>
<td>UCPD1 / UCPD2</td>
<td>UCPD global interrupt (combined with EXTI lines 32 &amp; 33)</td>
<td>0x0000_0060</td>
</tr>
<tr>
<td>9</td>
<td>16</td>
<td>settable</td>
<td>DMA_Channel1</td>
<td>DMA channel 1 interrupt</td>
<td>0x0000_0064</td>
</tr>
<tr>
<td>10</td>
<td>17</td>
<td>settable</td>
<td>DMA_Channel2_3</td>
<td>DMA channel 2 &amp; 3 interrupts</td>
<td>0x0000_0068</td>
</tr>
<tr>
<td>11</td>
<td>18</td>
<td>settable</td>
<td>DMA_Channel4_5_6_7 / DMAMUX</td>
<td>DMA channel 4, 5, 6 &amp; 7 and DMAMUX interrupts</td>
<td>0x0000_006C</td>
</tr>
<tr>
<td>12</td>
<td>19</td>
<td>settable</td>
<td>ADC_COMP</td>
<td>ADC and COMP interrupts (ADC combined with EXTI 17 &amp; 18)</td>
<td>0x0000_0070</td>
</tr>
<tr>
<td>13</td>
<td>20</td>
<td>settable</td>
<td>TIM1_BRK_UP_TRG_COM</td>
<td>TIM1 break, update, trigger and commutation interrupts</td>
<td>0x0000_0074</td>
</tr>
<tr>
<td>14</td>
<td>21</td>
<td>settable</td>
<td>TIM1_CC</td>
<td>TIM1 Capture Compare interrupt</td>
<td>0x0000_0078</td>
</tr>
<tr>
<td>15</td>
<td>22</td>
<td>settable</td>
<td>TIM2</td>
<td>TIM2 global interrupt</td>
<td>0x0000_007C</td>
</tr>
<tr>
<td>16</td>
<td>23</td>
<td>settable</td>
<td>TIM3</td>
<td>TIM3 global interrupt</td>
<td>0x0000_0080</td>
</tr>
<tr>
<td>17</td>
<td>24</td>
<td>settable</td>
<td>TIM6_DAC / LPTIM1</td>
<td>TIM6, LPTIM1 and DAC global interrupt</td>
<td>0x0000_0084</td>
</tr>
<tr>
<td>18</td>
<td>25</td>
<td>settable</td>
<td>TIM7 / LPTIM2</td>
<td>TIM7 and LPTIM2 global interrupt</td>
<td>0x0000_0088</td>
</tr>
<tr>
<td>19</td>
<td>26</td>
<td>settable</td>
<td>TIM14</td>
<td>TIM14 global interrupt</td>
<td>0x0000_008C</td>
</tr>
<tr>
<td>20</td>
<td>27</td>
<td>settable</td>
<td>TIM15</td>
<td>TIM15 global interrupt</td>
<td>0x0000_0090</td>
</tr>
<tr>
<td>21</td>
<td>28</td>
<td>settable</td>
<td>TIM16</td>
<td>TIM16 global interrupt</td>
<td>0x0000_0094</td>
</tr>
<tr>
<td>22</td>
<td>29</td>
<td>settable</td>
<td>TIM17</td>
<td>TIM17 global interrupt</td>
<td>0x0000_0098</td>
</tr>
<tr>
<td>23</td>
<td>30</td>
<td>settable</td>
<td>I2C1</td>
<td>I2C1 global interrupt (combined with EXTI 23)</td>
<td>0x0000_009C</td>
</tr>
<tr>
<td>24</td>
<td>31</td>
<td>settable</td>
<td>I2C2</td>
<td>I2C2 global interrupt</td>
<td>0x0000_00A0</td>
</tr>
</tbody>
</table>
### Table 48. Vector table(1) (continued)

<table>
<thead>
<tr>
<th>Position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>32</td>
<td>settable</td>
<td>SPI1</td>
<td>SPI1 global interrupt</td>
<td>0x0000_00A4</td>
</tr>
<tr>
<td>26</td>
<td>33</td>
<td>settable</td>
<td>SPI2</td>
<td>SPI2 global interrupt</td>
<td>0x0000_00A8</td>
</tr>
<tr>
<td>27</td>
<td>34</td>
<td>settable</td>
<td>USART1</td>
<td>USART1 global interrupt (combined with EXTI 25)</td>
<td>0x0000_00AC</td>
</tr>
<tr>
<td>28</td>
<td>35</td>
<td>settable</td>
<td>USART2</td>
<td>USART2 global interrupt (combined with EXTI 26)</td>
<td>0x0000_00B0</td>
</tr>
<tr>
<td>29</td>
<td>36</td>
<td>settable</td>
<td>USART3 / USART4 / LPUART1</td>
<td>USART3, USART4, and LPUART1 global interrupt (combined with EXTI 28)</td>
<td>0x0000_00B4</td>
</tr>
<tr>
<td>30</td>
<td>37</td>
<td>settable</td>
<td>CEC</td>
<td>CEC global interrupt (combined with EXTI 27)</td>
<td>0x0000_00B8</td>
</tr>
<tr>
<td>31</td>
<td>38</td>
<td>settable</td>
<td>AES / RNG</td>
<td>AES and RNG global interrupts</td>
<td>0x0000_00BC</td>
</tr>
</tbody>
</table>

1. The grayed cells correspond to the Cortex®-M0+ interrupts.
12 Extended interrupt and event controller (EXTI)

The Extended interrupt and event controller (EXTI) manages the CPU and system wakeup through configurable and direct event inputs (lines). It provides wakeup requests to the power control, and generates an interrupt request to the CPU NVIC and events to the CPU event input. For the CPU an additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wakeup requests allow the system to be woken up from Stop modes.

The interrupt request and event request generation can also be used in Run modes.

The EXTI also includes the EXTI I/O port mux.

12.1 EXTI main features

The EXTI main features are the following:

- System wakeup upon event on any input
- Wakeup flag and CPU interrupt generation for events not having a wakeup flag in their source peripheral
- Configurable events (from I/Os, peripherals not having an associated interrupt pending status bit, or peripherals generating a pulse)
  - Selectable active trigger edge
  - Independent rising and falling edge interrupt pending status bits
  - Individual interrupt and event generation mask, used for conditioning the CPU wakeup, interrupt and event generation
  - SW trigger possibility
- Direct events (from peripherals having an associated flag and interrupt pending status bit)
  - Fixed rising edge active trigger
  - No interrupt pending status bit in the EXTI
  - Individual interrupt and event generation mask for conditioning the CPU wakeup and event generation
  - No SW trigger possibility
- I/O port selector

12.2 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input trigger block, the masking block, and EXTI mux as shown in Figure 24.

The register block contains all the EXTI registers.

The event input trigger block provides an event input edge trigger logic.

The masking block provides the event input distribution to the different wakeup, interrupt and event outputs, and the masking of these.

The EXTI mux provides the I/O port selection on to the EXTI event signal.
Table 49. EXTI signal overview

<table>
<thead>
<tr>
<th>Signal name</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHB interface</td>
<td>I/O</td>
<td>EXTI register bus interface. When one event is configured to allow security, the AHB interface support secure accesses</td>
</tr>
<tr>
<td>hclk</td>
<td>I</td>
<td>AHB bus clock and EXTI system clock</td>
</tr>
<tr>
<td>Configurable event(y)</td>
<td>I</td>
<td>Asynchronous wakeup events from peripherals that do not have an associated interrupt and flag in the peripheral</td>
</tr>
<tr>
<td>Direct event(x)</td>
<td>I</td>
<td>Synchronous and asynchronous wakeup events from peripherals having an associated interrupt and flag in the peripheral</td>
</tr>
<tr>
<td>IOPort(n)</td>
<td>I</td>
<td>GPIO ports[15:0]</td>
</tr>
<tr>
<td>exti[15:0]</td>
<td>O</td>
<td>EXTI output port to trigger other IPs</td>
</tr>
<tr>
<td>it_exti_per(y)</td>
<td>O</td>
<td>Interrupts to the CPU associated with configurable event (y)</td>
</tr>
<tr>
<td>c_evt_exti</td>
<td>O</td>
<td>High-level sensitive event output for CPU synchronous to hclk</td>
</tr>
<tr>
<td>c_evt_rst</td>
<td>I</td>
<td>Asynchronous reset input to clear c_evt_exti</td>
</tr>
<tr>
<td>sys_wakeup</td>
<td>O</td>
<td>Asynchronous system wakeup request to PWR for ck_sys and hclk</td>
</tr>
<tr>
<td>c_wakeup</td>
<td>O</td>
<td>Wakeup request to PWR for CPU, synchronous to hclk</td>
</tr>
</tbody>
</table>

Table 50. EVG pin overview

<table>
<thead>
<tr>
<th>Pin name</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_fclk</td>
<td>I</td>
<td>CPU free-running clock</td>
</tr>
<tr>
<td>c_evt_in</td>
<td>I</td>
<td>High-level sensitive event input from EXTI, asynchronous to CPU clock</td>
</tr>
<tr>
<td>c_event</td>
<td>O</td>
<td>Event pulse, synchronous to CPU clock</td>
</tr>
<tr>
<td>c_evt_rst</td>
<td>O</td>
<td>Event reset signal, synchronous to CPU clock</td>
</tr>
</tbody>
</table>
12.2.1 EXTI connections between peripherals and CPU

The peripherals able to generate wakeup or interrupt events when the system is in Stop mode are connected to the EXTI.

- Peripheral wakeup signals that generate a pulse or that do not have an interrupt status bits in the peripheral, are connect to an EXTI configurable line. For these events the EXTI provides a status pending bit which requires to be cleared. It is the EXTI interrupt associated with the status bit that will interrupt the CPU.
- Peripheral interrupt and wakeup signals that have a status bit in the peripheral which requires to be cleared in the peripheral, are connected to an EXTI direct line. There is no status pending bit within the EXTI. The interrupt or wakeup is cleared by the CPU in the peripheral. It is the peripheral interrupt that will interrupt the CPU directly.
- All GPIO ports input to the EXTI multiplexer, allowing to select a port to wake up the system via a configurable event.

The EXTI configurable event interrupts are connected to the NVIC(a) of the CPU.

The dedicated EXTI/EVG CPU event is connected to the CPU rxev input.

The EXTI CPU wakeup signals are connected to the PWR block, and are used to wake up the system and CPU sub-system bus clocks.

12.3 EXTI functional description

Depending on the EXTI line type and wakeup target(s), different logic implementations are used. The applicable features and control or status registers are:

- rising and falling edge event enable through
  - EXTI rising trigger selection register (EXTI_RTSR1)
  - EXTI falling trigger selection register (EXTI_FTSR1)
- software trigger through EXTI software interrupt event register (EXTI_SWIER1)
- pending interrupt flagging through
  - EXTI rising edge pending register (EXTI_RPR1)
  - EXTI falling edge pending register (EXTI_FPR1)
  - EXTI external interrupt selection register (EXTI_EXTICR).x
- CPU wakeup and interrupt enable through
  - EXTI CPU wakeup with interrupt mask register (EXTI_IMR1)
  - EXTI CPU wakeup with interrupt mask register (EXTI_IMR2)
- CPU wakeup and event enable through
  - EXTI CPU wakeup with event mask register (EXTI_EMR1)
  - EXTI CPU wakeup with event mask register (EXTI_EMR2)
### Table 51. EXTI event input configurations and register control

<table>
<thead>
<tr>
<th>Event input type</th>
<th>Logic implementation</th>
<th>EXTI_RTSR1</th>
<th>EXTI_FTSR1</th>
<th>EXTI_SWIER1</th>
<th>EXTI_RFPR1</th>
<th>EXTI_IMRx</th>
<th>EXTI_EMRx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configurable</td>
<td>Configurable event input wakeup logic</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Direct</td>
<td>Direct event input wakeup logic</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

#### 12.3.1 EXTI configurable event input wakeup

*Figure 25* is a detailed representation of the logic associated with configurable event inputs which will wake up the CPU sub-system bus clocks and generate an EXTI pending flag and interrupt to the CPU and or a CPU wakeup event.

*Figure 25. Configurable event trigger logic CPU wakeup*

The software interrupt event register allows triggering configurable events by software, writing the corresponding register bit, irrespective of the edge selection setting.

The rising edge and falling edge selection registers allow to enable and select the configurable event active trigger edge or both edges.

The CPU has its dedicated interrupt mask register and a dedicated event mask registers. The enabled event allows generating an event on the CPU. All events for a CPU are OR-ed together into a single CPU event signal. The event pending registers (EXTI_RPR1 and EXTI_FPR1) is not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers, shared by the CPU. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to the CPU. The configurable event interrupts need to be acknowledged by software in the EXTI_RPR1 and/or EXTI_FPR1 registers.

When a CPU interrupt or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees the wakeup of the EXTI hclk clock before the asynchronous edge detection circuit is reset.
A detected configurable event interrupt pending request can be cleared by the CPU. The system cannot enter low-power modes as long as an interrupt pending request is active.

12.3.2 EXTI direct event input wakeup

Figure 26 is a detailed representation of the logic associated with direct event inputs waking up the system.

The direct events do not have an associated EXTI interrupt. The EXTI only wakes up the system and CPU sub-system clocks and may generate a CPU wakeup event. The peripheral synchronous interrupt, associated with the direct wakeup event wakes up the CPU.

The EXTI direct event is able to generate a CPU event. This CPU event wakes up the CPU. The CPU event may occur before the interrupt flag of the associated peripheral is set.

Figure 26. Direct event trigger logic CPU wakeup

12.3.3 EXTI mux

The EXTI mux allows selecting GPIOs as interrupts and wakeup. The GPIOs are connected via 16 EXTI mux lines to the first 16 EXTI events as configurable event. The selection of GPIO port as EXTI mux output is controlled through the EXTI external interrupt selection register (EXTI_EXTICRxs) register.
The EXTI lines (event inputs) are connected as shown in the following table.

### Table 52. EXTI line connections

<table>
<thead>
<tr>
<th>EXTI line</th>
<th>Line source</th>
<th>Line type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-15</td>
<td>GPIO</td>
<td>Configurable</td>
</tr>
<tr>
<td>16</td>
<td>PVD output</td>
<td>Configurable</td>
</tr>
<tr>
<td>17</td>
<td>COMP1 output</td>
<td>Configurable</td>
</tr>
<tr>
<td>18</td>
<td>COMP2 output</td>
<td>Configurable</td>
</tr>
<tr>
<td>19</td>
<td>RTC</td>
<td>Direct</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td>21</td>
<td>TAMP</td>
<td>Direct</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>I2C1 wakeup</td>
<td>Direct</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>USART1 wakeup</td>
<td>Direct</td>
</tr>
<tr>
<td>26</td>
<td>USART2 wakeup</td>
<td>Direct</td>
</tr>
<tr>
<td>27</td>
<td>CEC wakeup</td>
<td>Direct</td>
</tr>
<tr>
<td>28</td>
<td>LPUART1 wakeup</td>
<td>Direct</td>
</tr>
<tr>
<td>29</td>
<td>LPTIM1</td>
<td>Direct</td>
</tr>
<tr>
<td>30</td>
<td>LPTIM2</td>
<td>Direct</td>
</tr>
<tr>
<td>31</td>
<td>LSE_CSS</td>
<td>Direct</td>
</tr>
<tr>
<td>32</td>
<td>UCPD1 wakeup</td>
<td>Direct</td>
</tr>
<tr>
<td>33</td>
<td>UCPD2 wakeup</td>
<td>Direct</td>
</tr>
</tbody>
</table>
12.4 EXTI functional behavior

The direct event inputs are enabled in the respective peripheral generating the wakeup event. The configurable events are enabled by enabling at least one of the trigger edges. Once an event input is enabled, the generation of a CPU wakeup is conditioned by the CPU interrupt mask and CPU event mask.

<table>
<thead>
<tr>
<th>CPU interrupt enable EXTI_IMR.IMn</th>
<th>CPU event enable EXTI_EMR.EMn</th>
<th>Configurable event inputs EXTI_RPR.RPIFn</th>
<th>exti(n) interrupt(1)</th>
<th>CPU event</th>
<th>CPU wakeup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>No</td>
<td>Masked</td>
<td>Masked</td>
<td>Masked</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Status latched</td>
<td>Yes</td>
<td>Masked</td>
<td>Yes</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Status latched</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

1. The single exti(n) interrupt goes to the CPU. If no interrupt is required for CPU, the exti(n) interrupt must be masked in the CPU NVIC.

2. Only if CPU interrupt is enabled in EXTI_IMR.IMn.

For configurable event inputs, upon an edge on the event input, an event request is generated if that edge (rising or/and falling) is enabled. When the associated CPU interrupt is unmasked, the corresponding RPIFn and/or FPIFn bit is/are set in the EXTI_RPR or/and EXTI_FPR register, waking up the CPU subsystem and activating CPU interrupt signal. The RPIFn and/or FPIFn pending bit is cleared by writing 1 to it, which clears the CPU interrupt request.

For direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit in the EXTI. When the associated CPU interrupt is unmasked, the corresponding CPU subsystem is woken up. The CPU is woken up (interrupted) by the peripheral synchronous interrupt.

The CPU event must be unmasked to generate an event. Upon an enabled edge occurring on an event input, a CPU event pulse is generated. There is no event pending bit.

For the configurable event inputs, the software can generate an event request by setting the corresponding bit of the software interrupt/event register EXTI_SWIER1, which has the effect of a rising edge on the event input. The pending rising edge event flag is set in the EXTI_RPR1 register, irrespective of the EXTI_RTSR1 register setting.

12.5 EXTI registers

The EXTI register map is divided in the following sections:

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000 - 0x01C</td>
<td>General configurable event [31:0] configuration</td>
</tr>
</tbody>
</table>
All the registers can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit) access.

### 12.5.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x000  
Reset value: 0x0000 0000  
Contains only register bits for configurable events.

<table>
<thead>
<tr>
<th>Bits 31:19</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bits 18:0  | **RTx**: Rising trigger event configuration bit of configurable line x (x = 18 to 0)\(^{(1)}\)  
Each bit enables/disables the rising edge trigger for the event and interrupt on the corresponding line.  
0: Disable  
1: Enable  
The RT18 and RT17 bits are only available in STM32G071xx and STM32G081xx. They are reserved in STM32G031xx and STM32G041xx. |

1. The configurable lines are edge triggered, no glitch must be generated on these inputs.  
If a rising edge on the configurable line occurs during writing of the register, the associated pending bit is not set.  
Rising edge trigger can be set for a line with falling edge trigger enabled. In this case, both edges generate a trigger.

### 12.5.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x004  
Reset value: 0x0000 0000  
Contains only register bits for configurable events.

<table>
<thead>
<tr>
<th>Bits 31:19</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
Bits 31:19  Reserved, must be kept at reset value.

Bits 18:0  FTx: Falling trigger event configuration bit of configurable line x (x = 18 to 0)\(^{(1)}\).
Each bit enables/disables the falling edge trigger for the event and interrupt on the corresponding line.
0: Disable
1: Enable

The FT18 and FT17 bits are only available in STM32G071xx and STM32G081xx. They are reserved in STM32G031xx and STM32G041xx.

---

1. The configurable lines are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the configurable line occurs during writing of the register, the associated pending bit is not set. Falling edge trigger can be set for a line with rising edge trigger enabled. In this case, both edges generate a trigger.

**12.5.3 EXTI software interrupt event register (EXTI_SWIER1)**

Address offset: 0x008
Reset value: 0x0000 0000
Contains only register bits for configurable events.

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

Bits 31:19  Reserved, must be kept at reset value.

Bits 18:0  SWIx: Software rising edge event trigger on line x (x = 18 to 0)
Setting of any bit by software triggers a rising edge event on the corresponding line x, resulting in an interrupt, independently of EXTI_RTSR1 and EXTI_FTSR1 settings. The bits are automatically cleared by HW. Reading of any bit always returns 0.
0: No effect
1: Rising edge event generated on the corresponding line, followed by an interrupt

The SW18 and SW17 bits are only available in STM32G071xx and STM32G081xx. They are reserved in STM32G031xx and STM32G041xx.

---

**12.5.4 EXTI rising edge pending register (EXTI_RPR1)**

Address offset: 0x00C
Reset value: 0x0000 0000
Contains only register bits for configurable events.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td></td>
</tr>
</tbody>
</table>

---

RM0444 Rev 2  289/1234
12.5.5 EXTI falling edge pending register (EXTI_FPR1)

Address offset: 0x010
Reset value: 0x0000 0000
Contains only register bits for configurable events.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **RPIFx:** Rising edge event pending for configurable line x (x = 18 to 0)
Each bit is set upon a rising edge event generated by hardware or by software (through the EXTI_SWIER1 register) on the corresponding line. Each bit is cleared by writing 1 into it.
0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
The RPIF18 and RPIF17 bits are only available in STM32G071xx and STM32G081xx. They are reserved in STM32G031xx and STM32G041xx.

12.5.6 EXTI external interrupt selection register (EXTI_EXTICRx)

Address offset: 0x060 + 0x4 * (x - 1), (x = 1 to 4)
Reset value: 0x0000 0000
EXTIm fields contain only the number of bits in line with the nb_iopar configuration.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **FPIFx:** Falling edge event pending for configurable line x (x = 18 to 0)
Each bit is set upon a falling edge event generated by hardware or by software (through the EXTI_SWIER1 register) on the corresponding line. Each bit is cleared by writing 1 into it.
0: No falling edge trigger request occurred
1: Falling edge trigger request occurred
The FPIF18 and FPIF17 bits are only available in STM32G071xx and STM32G081xx. They are reserved in STM32G031xx and STM32G041xx.
12.5.7 EXTI CPU wakeup with interrupt mask register (EXTI_IMR1)

Address offset: 0x080 (EXTI_IMR1)

Reset value: 0xFFFF8 0000

Contains register bits for configurable events and direct events.

The reset value is set such as to, by default, enable interrupt from direct lines, and disable interrupt from configurable lines.
### 12.5.8 EXTI CPU wakeup with event mask register (EXTI_EMR1)

Address offset: 0x084 (EXTI_EMR1)

Reset value: 0x0000 0000
12.5.9 EXTI CPU wakeup with interrupt mask register (EXTI_IMR2)

Address offset: 0x090 (EXTI_IMR2)
Reset value: 0xFFFF FFFF
Contains register bits for configurable events and direct events.
This register is not available in STM32G031xx and STM32G041xx.

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

**Bits 31:25 EMx**: CPU wakeup with event generation mask on line x (x = 31 to 25)
Setting/clearing each bit unmasks/masks the CPU wakeup with event generation on the corresponding line.
0: wakeup with event generation masked
1: wakeup with event generation unmasked

The EM27 bit is only available in STM32G071xx and STM32G081xx. It is reserved in STM32G031xx and STM32G041xx.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **EM23**: CPU wakeup with event generation mask on line 23
Setting/clearing this bit unmasks/masks the CPU wakeup with event generation on the corresponding line.
0: wakeup with event generation masked
1: wakeup with event generation unmasked

Bit 22 Reserved, must be kept at reset value.

Bit 21 **EM21**: CPU wakeup with event generation mask on line 21
Setting/clearing this bit unmasks/masks the CPU wakeup with event generation on the corresponding line.
0: wakeup with event generation masked
1: wakeup with event generation unmasked

Bit 20 Reserved, must be kept at reset value.

Bits 19:0 **EMx**: CPU wakeup with event generation mask on line x (x = 19 to 0)
Setting/clearing each bit unmasks/masks the CPU wakeup with event generation on the corresponding line.
0: wakeup with event generation masked
1: wakeup with event generation unmasked

The EM18 and EM17 bits are only available in STM32G071xx and STM32G081xx. They are reserved in STM32G031xx and STM32G041xx.
12.5.10 EXTI CPU wakeup with event mask register (EXTI_EMR2)

Address offset: 0x094
Reset value: 0x0000 0000

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

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

Bit 1 **EM33**: CPU wakeup with event generation mask on line 33
Setting/clearing each bit unmasksmasks the CPU wakeup with event generation on the corresponding line.
0: wakeup with event generation masked
1: wakeup with event generation unmasked

Bit 0 **EM32**: CPU wakeup with event generation mask on line 32
Setting/clearing each bit unmasksmasks the CPU wakeup with event generation on the corresponding line.
0: wakeup with event generation masked
1: wakeup with event generation unmasked

1. The reset value for configurable lines is set to 0 in order to disable the interrupt by default.
2. The reset value for direct lines is set to 1 in order to enable the interrupt by default.

12.5.11 EXTI register map

The following table gives the EXTI register map and the reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>EXTI_RTSR1</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
</tr>
</tbody>
</table>

Reset value

Table 55. EXTI controller register map and reset values

294/1234
### Table 55. EXTI controller register map and reset values (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x060</td>
<td>EXTI_EXTICR1</td>
<td>EXTI3[7:0]</td>
<td>EXTI1[7:0]</td>
<td>EXTI0[7:0]</td>
<td>EXTI[7:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Offset</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
</tr>
<tr>
<td>0x064</td>
<td>EXTI_EXTICR2</td>
<td>EXTI7[7:0]</td>
<td>EXTI6[7:0]</td>
<td>EXTI5[7:0]</td>
<td>EXTI4[7:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Offset</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
</tr>
<tr>
<td>0x068</td>
<td>EXTI_EXTICR3</td>
<td>EXTI11[7:0]</td>
<td>EXTI10[7:0]</td>
<td>EXTI9[7:0]</td>
<td>EXTI8[7:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Offset</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
</tr>
<tr>
<td>0x06C</td>
<td>EXTI_EXTICR4</td>
<td>EXTI15[7:0]</td>
<td>EXTI14[7:0]</td>
<td>EXTI13[7:0]</td>
<td>EXTI12[7:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Offset</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
</tr>
<tr>
<td>0x070-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x07C</td>
<td></td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
</tr>
</tbody>
</table>

Refer to Section 2.2 on page 54 for the register boundary addresses.
13 Cyclic redundancy check calculation unit (CRC)

13.1 Introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial. Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

13.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  \[X^{32} + X^{26} + X^{23} + X^{16} + X^{12} + X^{11} + X^{10} + 1 + X^7 + X^5 + X^4 + X^2 + 1\]
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-, 16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data
13.3 CRC functional description

13.3.1 CRC block diagram

Figure 28. CRC calculation unit block diagram

13.3.2 CRC internal signals

Table 56. CRC internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crc_hclk</td>
<td>Digital input</td>
<td>AHB clock</td>
</tr>
</tbody>
</table>

13.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit access is allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32-bit
- 2 AHB clock cycles for 16-bit
- 1 AHB clock cycles for 8-bit

An input buffer allows to immediately write a second data without waiting for any wait states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.
The input data can be reversed, to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example: input data 0x1A2B3C4D is used for CRC calculation as:
- 0x58D43CB2 with bit-reversal done by byte
- 0xD458B23C with bit-reversal done by half-word
- 0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

The operation is done at bit level: for example, output data 0x11223344 is converted into 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

**Polynomial programmability**

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size cannot be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.
13.4 CRC registers

13.4.1 CRC data register (CRC_DR)
Address offset: 0x00
Reset value: 0xFFFF FFFF

<table>
<thead>
<tr>
<th>Bits 31:0 DR[31:0]: Data register bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>This register is used to write new data to the CRC calculator.</td>
</tr>
<tr>
<td>It holds the previous CRC calculation result when it is read.</td>
</tr>
<tr>
<td>If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 31:0 IDR[31:0]: General-purpose 32-bit data register bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>These bits can be used as a temporary storage location for four bytes.</td>
</tr>
<tr>
<td>This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register</td>
</tr>
</tbody>
</table>

13.4.2 CRC independent data register (CRC_IDR)
Address offset: 0x04
Reset value: 0x0000 0000
### 13.4.3 CRC control register (CRC_CR)

**Address offset:** 0x08  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:8</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
</tbody>
</table>

- **Bit 7** REV_OUT: Reverse output data  
  This bit controls the reversal of the bit order of the output data.  
  0: Bit order not affected  
  1: Bit-reversed output format

- **Bits 6:5** REV_IN[1:0]: Reverse input data  
  These bits control the reversal of the bit order of the input data  
  00: Bit order not affected  
  01: Bit reversal done by byte  
  10: Bit reversal done by half-word  
  11: Bit reversal done by word

- **Bits 4:3** POLYSIZE[1:0]: Polynomial size  
  These bits control the size of the polynomial.  
  00: 32 bit polynomial  
  01: 16 bit polynomial  
  10: 8 bit polynomial  
  11: 7 bit polynomial

- **Bits 2:1** Reserved, must be kept at reset value.

- **Bit 0** RESET: RESET bit  
  This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware

### 13.4.4 CRC initial value (CRC_INIT)

**Address offset:** 0x10  
**Reset value:** 0xFFFF FFFF

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
</tbody>
</table>
13.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14
Reset value: 0x04C1 1DB7

Bits 31:0 CRC_INIT[31:0]: Programmable initial CRC value
This register is used to write the CRC initial value.

Bits 31:0 POL[31:0]: Programmable polynomial
This register is used to write the coefficients of the polynomial to be used for CRC calculation.
If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

13.4.6 CRC register map

Table 57. CRC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 31:0</th>
<th>Bits 28:16</th>
<th>Bits 15:0</th>
<th>Bits 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CRC_DR</td>
<td>DR[31:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>1 1 1 1 1</td>
<td>1 1 1 1 1 1</td>
<td>1 1 1 1 1</td>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>0x04</td>
<td>CRC_IDR</td>
<td>IDR[31:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0x08</td>
<td>CRC_CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0x10</td>
<td>CRC_INIT</td>
<td>CRC_INIT[31:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>1 1 1 1 1 1</td>
<td>1 1 1 1 1</td>
<td>1 1 1 1 1</td>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>0x14</td>
<td>CRC_POL</td>
<td>Polynomial coefficients</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 1 1 0 1 1 0 1 1 0 1 1 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.2 on page 54 for the register boundary addresses.
14 Analog-to-digital converter (ADC)

14.1 Introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 19 multiplexed channels allowing it to measure signals from 16 external and 3 internal sources. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The analog watchdog feature allows the application to detect if the input voltage goes outside the user-defined higher or lower thresholds.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

A built-in hardware oversampler allows to improve analog performances while off-loading the related computational burden from the CPU.
14.2 ADC main features

- High performance
  - 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
  - ADC conversion time: 0.4 µs for 12-bit resolution (2.5Msps), faster conversion times can be obtained by lowering resolution.
  - Self-calibration
  - Programmable sampling time
  - Data alignment with built-in data coherency
  - DMA support

- Low-power
  - Application can reduce PCLK frequency for low-power operation while still keeping optimum ADC performance. For example, 0.4 µs conversion time is kept, whatever the frequency of PCLK
  - Wait mode: prevents ADC overrun in applications with low frequency PCLK
  - Auto off mode: ADC is automatically powered off except during the active conversion phase. This dramatically reduces the power consumption of the ADC.

- Analog input channels
  - 16 external analog inputs
  - 1 channel for internal temperature sensor (VTS)
  - 1 channel for internal reference voltage (VREFINT)
  - 1 channel for monitoring external VBAT power supply pin

- Start-of-conversion can be initiated:
  - By software
  - By hardware triggers with configurable polarity (timer events or GPIO input events)

- Conversion modes
  - Can convert a single channel or can scan a sequence of channels.
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode

- Interrupt generation at the end of sampling, end of conversion, end of sequence conversion, and in case of analog watchdog or overrun events

- Analog watchdog

- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256x
  - Programmable data shift up to 8-bits

- ADC supply requirements: 1.62 to 3.6 V
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

Figure 29 shows the block diagram of the ADC.
14.3 ADC functional description

Figure 29 shows the ADC block diagram and Table 59 gives the ADC pin description.

Figure 29. ADC block diagram

14.3.1 ADC pins and internal signals

Table 58. ADC internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRGx</td>
<td>Input</td>
<td>ADC conversion triggers</td>
</tr>
<tr>
<td>$V_{TS}$</td>
<td>Input</td>
<td>Internal temperature sensor output voltage</td>
</tr>
<tr>
<td>$V_{REFINT}$</td>
<td>Input</td>
<td>Internal voltage reference output voltage</td>
</tr>
<tr>
<td>$V_{BAT/3}$</td>
<td>Input</td>
<td>$V_{BAT}$ pin input voltage divided by 3</td>
</tr>
<tr>
<td>ADC_AWDx_OUT</td>
<td>Output</td>
<td>Internal analog watchdog output signal connected to on-chip timers ($x =$ Analog watchdog number = 1, 2, 3)</td>
</tr>
</tbody>
</table>
The ADC has a specific internal voltage regulator which must be enabled and stable before using the ADC.

The ADC internal voltage regulator can be enabled by setting ADVREGEN bit to 1 in the ADC_CR register. The software must wait for the ADC voltage regulator startup time (tADCVREG_SETUP) before launching a calibration or enabling the ADC. This delay must be managed by software (for details on tADCVREG_SETUP, refer to the device datasheet).

After ADC operations are complete, the ADC is disabled (ADEN=0). It is then possible to save additional power by disabling the ADC voltage regulator (refer to Section : ADC voltage regulator disable sequence).

**Note:** When the internal voltage regulator is disabled, the internal analog calibration is kept.

### Analog reference from the power control unit

The internal ADC voltage regulator internally uses an analog reference delivered by the power control unit through a buffer. This buffer is always enabled when the main voltage regulator of the power control unit operates in normal Run mode (refer to Reset and clock control and power control sections).

If the Main voltage regulator enters low-power mode (such as Low-power run mode), this buffer is disabled and the ADC cannot be used.

**ADC Voltage regulator enable sequence**

To enable the ADC voltage regulator, set ADVREGEN bit to 1 in ADC_CR register.

**ADC voltage regulator disable sequence**

To disable the ADC voltage regulator, follow the sequence below:
1. Make sure that the ADC is disabled (ADEN=0)
2. Clear ADVREGEN bit in ADC_CR register.

---

Table 59. ADC input/output pins

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDDA</td>
<td>Input, analog power supply</td>
<td>Analog power supply and positive reference voltage for the ADC, VDDA ≥ VDD</td>
</tr>
<tr>
<td>VSSA</td>
<td>Input, analog supply ground</td>
<td>Ground for analog power supply. Must be at VSS potential</td>
</tr>
<tr>
<td>VIN[x]</td>
<td>Input analog channels</td>
<td>Connected to VREF, or to external channels, ADC_IN/x</td>
</tr>
<tr>
<td>ADC_INx</td>
<td>Analog input signals</td>
<td>16 analog input channels</td>
</tr>
</tbody>
</table>
14.3.3 Calibration (ADCAL)

The ADC has a calibration feature. During the procedure, the ADC calculates a calibration factor which is internally applied to the ADC until the next ADC power-off. The application must not use the ADC during calibration and must wait until it is complete.

Calibration should be performed before starting A/D conversion. It removes the offset error which may vary from chip to chip due to process variation.

The calibration is initiated by software by setting bit ADCAL=1. Calibration can only be initiated when the ADC voltage regulator is enabled (ADVREGEN=1, and tADCVREG_SETUP has elapsed) and the ADC is disabled (when ADEN=0). ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. After this, the calibration factor can be read from the ADC_DR register (from bits 6 to 0).

The internal analog calibration is kept if the ADC is disabled (ADEN=0). When the ADC operating conditions change (VDDA changes are the main contributor to ADC offset variations and temperature change to a lesser extend), it is recommended to re-run a calibration cycle.

The calibration factor is lost in the following cases:

- The power supply is removed from the ADC (for example when the product enters STANDBY or VBAT mode)
- The ADC peripheral is reset.

The calibration factor is lost each time power is removed from the ADC (for example when the product enters Standby or VBAT mode). Still, it is possible to save and restore the calibration factor by software to save time when re-starting the ADC (as long as temperature and voltage are stable during the ADC power-down).

The calibration factor can be written if the ADC is enabled but not converting (ADEN=1 and ADSTART=0). Then, at the next start of conversion, the calibration factor will automatically be injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion.

Software calibration procedure

1. Ensure that ADEN=0, ADVREGEN=1 and DMAEN=0.
2. Set ADCAL=1.
3. Wait until ADCAL=0 (or until EOCAL=1). This can be handled by interrupt if the interrupt is enabled by setting the EOCALIE bit in the ADC_IER register
4. The calibration factor can be read from bits 6:0 of ADC_DR or ADC_CALFACT registers.
Figure 30. ADC calibration

Calibration factor forcing Software Procedure

1. Ensure that ADEN= 1 and ADSTART =0 (ADC started with no conversion ongoing)
2. Write ADC_CALFACT with the saved calibration factor
3. The calibration factor will be used as soon as a new conversion will be launched.

Figure 31. Calibration factor forcing

14.3.4 ADC on-off control (ADEN, ADDIS, ADRDY)

At power-up, the ADC is disabled and put in power-down mode (ADEN=0).

As shown in Figure 32, the ADC needs a stabilization time of t_{STAB} before it starts converting accurately.

Two control bits are used to enable or disable the ADC:

- Set ADEN=1 to enable the ADC. The ADRDY flag is set as soon as the ADC is ready for operation.
- Set ADDIS=1 to disable the ADC and put the ADC in power down mode. The ADEN and ADDIS bits are then automatically cleared by hardware as soon as the ADC is fully disabled.

Conversion can then start either by setting ADSTART to 1 (refer to Section 14.4: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN) on page 317) or when an external trigger event occurs if triggers are enabled.
Follow this procedure to enable the ADC:

1. Clear the ADRDY bit in ADC_ISR register by programming this bit to 1.
2. Set ADEN=1 in the ADC_CR register.
3. Wait until ADRDY=1 in the ADC_ISR register (ADRDY is set after the ADC startup time). This can be handled by interrupt if the interrupt is enabled by setting the ADRDYIE bit in the ADC_IER register.

Follow this procedure to disable the ADC:

1. Check that ADSTART=0 in the ADC_CR register to ensure that no conversion is ongoing. If required, stop any ongoing conversion by writing 1 to the ADSTP bit in the ADC_CR register and waiting until this bit is read at 0.
2. Set ADDIS=1 in the ADC_CR register.
3. If required by the application, wait until ADEN=0 in the ADC_CR register, indicating that the ADC is fully disabled (ADDIS is automatically reset once ADEN=0).
4. Clear the ADRDY bit in ADC_ISR register by programming this bit to 1 (optional).

**Figure 32. Enabling/disabling the ADC**

![Diagram showing enabling/disabling the ADC](image)

**Note:** In Auto-off mode (AUTOFF=1) the power-on/off phases are performed automatically, by hardware and the ADRDY flag is not set.
14.3.5 ADC clock (CKMODE, PRESC[3:0])

The ADC has a dual clock-domain architecture, so that the ADC can be fed with a clock (ADC asynchronous clock) independent from the APB clock (PCLK).

Figure 33. ADC clock scheme

1. Refer to Section Reset and clock control (RCC) for how PCLK and ADC asynchronous clock are enabled.

The input clock of the analog ADC can be selected between two different clock sources (see Figure 33: ADC clock scheme to see how PCLK and the ADC asynchronous clock are enabled):

a) The ADC clock can be a specific clock source, named “ADC asynchronous clock” which is independent and asynchronous with the APB clock.
   Refer to RCC Section for more information on generating this clock source.
   To select this scheme, bits CKMODE[1:0] of the ADC_CFGR2 register must be reset.

b) The ADC clock can be derived from the APB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4) according to bits CKMODE[1:0].
   To select this scheme, bits CKMODE[1:0] of the ADC_CFGR2 register must be different from “00”.

In option a), the generated ADC clock can eventually be divided by a prescaler (1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128, 256) when programming the bits PRESC[3:0] in the ADC_CCR register).

Option a) has the advantage of reaching the maximum ADC clock frequency whatever the APB clock scheme selected.

Option b) has the advantage of bypassing the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).
Caution: When selecting CKMODE[1:0]=11 (PCLK divided by 1), the user must ensure that PCLK has a 50% duty cycle. This is done by selecting a system clock with a 50% duty cycle and configuring the APB prescaler in bypass modes in the RCC (refer to there Reset and clock controller section). In case of an internal source clock, this implies only that the AHB and APB prescalers do not divide the clock.

Note: Refer to the device datasheet for the maximum ADC_CLK frequency.

<table>
<thead>
<tr>
<th>ADC clock source</th>
<th>CKMODE[1:0]</th>
<th>Latency between the trigger event and the start of conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSI16, SYSCLK, or PLLPCLK(1)</td>
<td>00</td>
<td>Latency is not deterministic (jitter)</td>
</tr>
<tr>
<td>PCLK divided by 2</td>
<td>01</td>
<td>Latency is deterministic (no jitter) and equal to 2.75 ADC clock cycles</td>
</tr>
<tr>
<td>PCLK divided by 4</td>
<td>10</td>
<td>Latency is deterministic (no jitter) and equal to 2.625 ADC clock cycles</td>
</tr>
<tr>
<td>PCLK divided by 1</td>
<td>11</td>
<td>Latency is deterministic (no jitter) and equal to 3 ADC clock cycles</td>
</tr>
</tbody>
</table>

1. Selected with ADCSEL bitfield of the RCC_CCIPR register
14.3.6 ADC connectivity

ADC inputs are connected to the external channels as well as internal sources as described in Figure 34.

Figure 34. ADC connectivity
14.3.7 Configuring the ADC

Software must write to the ADCAL and ADEN bits in the ADC_CR register if the ADC is disabled (ADEN must be 0).

Software must only write to the ADSTART and ADDIS bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN = 1 and ADDIS = 0).

For all the other control bits in the ADC_IER, ADC_CFGRI, ADC_SMPR, ADC_CHSELR and ADC_CCR registers, refer to the description of the corresponding control bit in Section 14.12: ADC registers.

ADC_AWDTRx registers can be modified when conversion is ongoing.

Software must only write to the ADSTP bit in the ADC_CR register if the ADC is enabled (and possibly converting) and there is no pending request to disable the ADC (ADSTART = 1 and ADDIS = 0).

Note: There is no hardware protection preventing software from making write operations forbidden by the above rules. If such a forbidden write access occurs, the ADC may enter an undefined state. To recover correct operation in this case, the ADC must be disabled (clear ADEN=0 and all the bits in the ADC_CR register).

14.3.8 Channel selection (CHSEL, SCANDIR, CHSELRMOD)

There are up to 19 multiplexed channels:

- 16 analog inputs from GPIO pins (ADC_INx)
- 3 internal analog inputs (Temperature Sensor, Internal Reference Voltage, VBAT channel)

It is possible to convert a single channel or a sequence of channels.

The sequence of the channels to be converted can be programmed in the ADC_CHSELR channel selection register: each analog input channel has a dedicated selection bit (CHSELRx).

The ADC scan sequencer can be used in two different modes:

- Sequencer not fully configurable:
  - The order in which the channels are scanned is defined by the channel number (CHSELRMOD bit must be cleared in ADC_CFGRI register):
    - Sequence length configured through CHSELx bits in ADC_CHSELR register
    - Sequence direction: the channels are scanned in a forward direction (from the lowest to the highest channel number) or backward direction (from the highest to the lowest channel number) depending on the value of SCANDIR bit (SCANDIR=0: forward scan, SCANDIR=1: backward scan)
Any channel can belong to these sequences
- Sequencer fully configurable
  - The CHSELRMOD bit is set in ADC_CFGR1 register.
- Sequencer length is up to 8 channels
- The order in which the channels are scanned is independent from the channel number. Any order can be configured through SQ1[3:0] to SQ8[3:0] bits in ADC_CHSELR register.
- Only channel 0 to channel 14 can be selected in this sequence
- If the sequencer detects SQx[3:0] = 0b1111, the following SQx[3:0] registers are ignored.
- If no 0b1111 is programmed in SQx[3:0], the sequencer will scan full eight channels.

After programming ADC CHSELR, SCANDIR and CHSELRMOD bits, it is mandatory to wait for CCRDY flag before starting conversions. It indicates that the new channel setting has been applied. If a new configuration is required, the CCRDY flag must be cleared prior to starting the conversion.

The software is allowed to program the CHSEL, SCANDIR, CHSELRMOD bits only when ADSTART bit is cleared to 0 (which ensures that no conversion is ongoing).

**Temperature sensor, VREFINT and VBAT internal channels**

The temperature sensor is connected to channel ADC VIN[12].

The internal voltage reference VREFINT is connected to channel ADC VIN[13].

The VBAT channel is connected to ADC VIN[14] channel.

### 14.3.9 Programmable sampling time (SMPx[2:0])

Before starting a conversion, the ADC needs to establish a direct connection between the voltage source to be measured and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the sample and hold capacitor to the input voltage level.

Having a programmable sampling time allows to trim the conversion speed according to the input resistance of the input voltage source.

The ADC samples the input voltage for a number of ADC clock cycles that can be modified using the SMP1[2:0] and SMP2[2:0] bits in the ADC_SMPR register.

Each channel can choose one out of two sampling times configured in SMP1[2:0] and SMP2[2:0] bitfields, through SMPSELx bits in ADC_SMPR register.

The total conversion time is calculated as follows:

\[ t_{\text{CONV}} = \text{Sampling time} + 12.5 \times \text{ADC clock cycles} \]

**Example:**

With ADC_CLK = 16 MHz and a sampling time of 1.5 ADC clock cycles:

\[ t_{\text{CONV}} = 1.5 + 12.5 = 14 \text{ ADC clock cycles} = 0.875 \mu s \]

The ADC indicates the end of the sampling phase by setting the EOSMP flag.
14.3.10 Single conversion mode (CONT=0)

In Single conversion mode, the ADC performs a single sequence of conversions, converting all the channels once. This mode is selected when CONT=0 in the ADC_CFGR1 register. Conversion is started by either:
- Setting the ADSTART bit in the ADC_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:
- The converted data are stored in the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:
- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then the ADC stops until a new external trigger event occurs or the ADSTART bit is set again.

Note: To convert a single channel, program a sequence with a length of 1.

14.3.11 Continuous conversion mode (CONT=1)

In continuous conversion mode, when a software or hardware trigger event occurs, the ADC performs a sequence of conversions, converting all the channels once and then automatically re-starts and continuously performs the same sequence of conversions. This mode is selected when CONT=1 in the ADC_CFGR1 register. Conversion is started by either:
- Setting the ADSTART bit in the ADC_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:
- The converted data are stored in the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:
- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN=1 and CONT=1.
14.3.12 Starting conversions (ADSTART)

Software starts ADC conversions by setting ADSTART=1.

When ADSTART is set, the conversion:
- Starts immediately if EXTEN = 00 (software trigger)
- At the next active edge of the selected hardware trigger if EXTEN ≠ 00

The ADSTART bit is also used to indicate whether an ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART=0, indicating that the ADC is idle.

The ADSTART bit is cleared by hardware:
- In single mode with software trigger (CONT=0, EXTEN=00)
  - At any end of conversion sequence (EOS=1)
- In discontinuous mode with software trigger (CONT=0, DISCEN=1, EXTEN=00)
  - At end of conversion (EOC=1)
- In all cases (CONT=x, EXTEN=XX)
  - After execution of the ADSTP procedure invoked by software (see Section 14.3.14: Stopping an ongoing conversion (ADSTP) on page 316)

Note: In continuous mode (CONT=1), the ADSTART bit is not cleared by hardware when the EOS flag is set because the sequence is automatically relaunched.

When hardware trigger is selected in single mode (CONT=0 and EXTEN = 01), ADSTART is not cleared by hardware when the EOS flag is set. This avoids the need for software having to set the ADSTART bit again and ensures the next trigger event is not missed.

After changing channel selection configuration (by programming ADC_CHSELR register or changing CHSELRMOD or SCANDIR), it is mandatory to wait until CCRDY flag is asserted before asserting ADSTART, otherwise the value written to ADSTART will be ignored.

14.3.13 Timings

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

\[ t_{CONV} = t_{SMPL} + t_{SAR} = \begin{cases} 1.5 \ \text{min} + 12.5 \ \text{12bit} \times t_{ADC_CLK} \\ 42.9 \ \text{ns} \ \text{min} + 357.1 \ \text{ns} \ \text{12bit} = 0.400 \ \mu \text{s} \ \text{min} (\text{for } f_{ADC_CLK} = 35 \ \text{MHz}) \end{cases} \]

Figure 35. Analog to digital conversion time

<table>
<thead>
<tr>
<th>ADC state</th>
<th>Analog channel</th>
<th>RDY</th>
<th>SAMPLING CH(N)</th>
<th>CONVERTING CH(N)</th>
<th>SAMPLING CH(N+1)</th>
<th>CH(N+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal S/H</td>
<td>Sample AIN(N+1)</td>
<td>Hold AIN(N)</td>
<td>( t_{SMPL}^{(1)} )</td>
<td>( t_{SAR}^{(2)} )</td>
<td>Sample AIN(N+1)</td>
<td></td>
</tr>
<tr>
<td>ADSTART</td>
<td>set by SW</td>
<td>( t_{SMPL}^{(1)} )</td>
<td>cleared by SW</td>
<td>set by HW</td>
<td>cleared by SW</td>
<td></td>
</tr>
<tr>
<td>EOSMP</td>
<td>set by HW</td>
<td>( t_{SMPL}^{(1)} )</td>
<td>cleared by SW</td>
<td>set by HW</td>
<td>cleared by SW</td>
<td></td>
</tr>
<tr>
<td>EOC</td>
<td></td>
<td>( t_{SAR}^{(2)} )</td>
<td>cleared by SW</td>
<td>set by HW</td>
<td>cleared by SW</td>
<td></td>
</tr>
<tr>
<td>ADC_DR</td>
<td></td>
<td>DATA N-1</td>
<td></td>
<td>DATA N</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1) \( t_{SMPL} \) depends on SMP[2:0]
(2) \( t_{SAR} \) depends on RES[2:0]
1. EXTEN =00 or EXTEN ≠ 00
2. Trigger latency (refer to datasheet for more details)
3. ADC_DR register write latency (refer to datasheet for more details)

### 14.3.14 Stopping an ongoing conversion (ADSTP)

The software can decide to stop any ongoing conversions by setting ADSTP=1 in the ADC_CR register.

This will reset the ADC operation and the ADC will be idle, ready for a new operation.

When the ADSTP bit is set by software, any ongoing conversion is aborted and the result is discarded (ADC_DR register is not updated with the current conversion).

The scan sequence is also aborted and reset (meaning that restarting the ADC would restart a new sequence).

Once this procedure is complete, the ADSTP and ADSTART bits are both cleared by hardware and the software must wait until ADSTART=0 before starting new conversions.

---

**Figure 36. ADC conversion timings**

**Figure 37. Stopping an ongoing conversion**
14.4 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)

A conversion or a sequence of conversion can be triggered either by software or by an external event (for example timer capture). If the EXTEN[1:0] control bits are not equal to “0b00”, then external events are able to trigger a conversion with the selected polarity. The trigger selection is effective once software has set bit ADSTART=1.

Any hardware triggers which occur while a conversion is ongoing are ignored. If bit ADSTART=0, any hardware triggers which occur are ignored.

*Table 61* provides the correspondence between the EXTEN[1:0] values and the trigger polarity.

<table>
<thead>
<tr>
<th>Source</th>
<th>EXTEN[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger detection disabled</td>
<td>00</td>
</tr>
<tr>
<td>Detection on rising edge</td>
<td>01</td>
</tr>
<tr>
<td>Detection on falling edge</td>
<td>10</td>
</tr>
<tr>
<td>Detection on both rising and falling edges</td>
<td>11</td>
</tr>
</tbody>
</table>

*Note:* The polarity of the external trigger can be changed only when the ADC is not converting (ADSTART= 0).

The EXTSEL[2:0] control bits are used to select which of 8 possible events can trigger conversions.

*Table 62* gives the possible external trigger for regular conversion.

Software source trigger events can be generated by setting the ADSTART bit in the ADC_CR register.

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>EXTSEL[2:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRG0</td>
<td>TIM1_TRGO2</td>
<td>000</td>
</tr>
<tr>
<td>TRG1</td>
<td>TIM1_CC4</td>
<td>001</td>
</tr>
<tr>
<td>TRG2</td>
<td>TIM2_TRGO</td>
<td>010</td>
</tr>
<tr>
<td>TRG3</td>
<td>TIM3_TRGO</td>
<td>011</td>
</tr>
<tr>
<td>TRG4</td>
<td>TIM15_TRGO</td>
<td>100</td>
</tr>
<tr>
<td>TRG5</td>
<td>TIM6_TRGO</td>
<td>101</td>
</tr>
<tr>
<td>TRG6</td>
<td>Reserved</td>
<td>110</td>
</tr>
<tr>
<td>TRG7</td>
<td>EXTI line 11</td>
<td>111</td>
</tr>
</tbody>
</table>

*Note:* The trigger selection can be changed only when the ADC is not converting (ADSTART= 0).
14.4.1 Discontinuous mode (DISCEN)

This mode is enabled by setting the DISCEN bit in the ADC_CFGR1 register.

In this mode (DISCEN=1), a hardware or software trigger event is required to start each conversion defined in the sequence. On the contrary, if DISCEN=0, a single hardware or software trigger event successively starts all the conversions defined in the sequence.

Example:
- DISCEN=1, channels to be converted = 0, 3, 7, 10
  - 1st trigger: channel 0 is converted and an EOC event is generated
  - 2nd trigger: channel 3 is converted and an EOC event is generated
  - 3rd trigger: channel 7 is converted and an EOC event is generated
  - 4th trigger: channel 10 is converted and both EOC and EOS events are generated.
  - 5th trigger: channel 0 is converted an EOC event is generated
  - 6th trigger: channel 3 is converted and an EOC event is generated
  - ...

- DISCEN=0, channels to be converted = 0, 3, 7, 10
  - 1st trigger: the complete sequence is converted: channel 0, then 3, 7 and 10. Each conversion generates an EOC event and the last one also generates an EOS event.
  - Any subsequent trigger events will restart the complete sequence.

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN=1 and CONT=1.

14.4.2 Programmable resolution (RES) - fast conversion mode

It is possible to obtain faster conversion times ($t_{SAR}$) by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] bits in the ADC_CFGR1 register. Lower resolution allows faster conversion times for applications where high data precision is not required.

Note: The RES[1:0] bit must only be changed when the ADEN bit is reset.

The result of the conversion is always 12 bits wide and any unused LSB bits are read as zeros.

Lower resolution reduces the conversion time needed for the successive approximation steps as shown in Table 63.
14.4.3 End of conversion, end of sampling phase (EOC, EOSMP flags)

The ADC indicates each end of conversion (EOC) event.

The ADC sets the EOC flag in the ADC_ISR register as soon as a new conversion data result is available in the ADC_DR register. An interrupt can be generated if the EOCIE bit is set in the ADC_IER register. The EOC flag is cleared by software either by writing 1 to it, or by reading the ADC_DR register.

The ADC also indicates the end of sampling phase by setting the EOSMP flag in the ADC_ISR register. The EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if the EOSMPIE bit is set in the ADC_IER register.

The aim of this interrupt is to allow the processing to be synchronized with the conversions. Typically, an analog multiplexer can be accessed in hidden time during the conversion phase, so that the multiplexer is positioned when the next sampling starts.

Note: As there is only a very short time left between the end of the sampling and the end of the conversion, it is recommended to use polling or a WFE instruction rather than an interrupt and a WFI instruction.

### Table 63. $t_{\text{SAR}}$ timings depending on resolution

<table>
<thead>
<tr>
<th>RES[1:0] bits</th>
<th>$t_{\text{SAR}}$ (ADC clock cycles)</th>
<th>$t_{\text{SAR}}$ (ns) at $f_{\text{ADC}} = 35 \text{ MHz}$</th>
<th>$t_{\text{SMPL}}$ (min) (ADC clock cycles)</th>
<th>$t_{\text{CONV}}$ (ADC clock cycles) (with min. $t_{\text{SMPL}}$)</th>
<th>$t_{\text{CONV}}$ (ns) at $f_{\text{ADC}} = 35 \text{ MHz}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12.5</td>
<td>357</td>
<td>1.5</td>
<td>14</td>
<td>400</td>
</tr>
<tr>
<td>10</td>
<td>10.5</td>
<td>300</td>
<td>1.5</td>
<td>12</td>
<td>343</td>
</tr>
<tr>
<td>8</td>
<td>8.5</td>
<td>243</td>
<td>1.5</td>
<td>10</td>
<td>286</td>
</tr>
<tr>
<td>6</td>
<td>6.5</td>
<td>186</td>
<td>1.5</td>
<td>8</td>
<td>229</td>
</tr>
</tbody>
</table>

**Table 63. $t_{\text{SAR}}$ timings depending on resolution**

RES[1:0] refers to the resolution bits in the ADC configuration register. $t_{\text{SAR}}$ is the settling time after the sample and hold stage, $t_{\text{SMPL}}$ is the minimum sample time, and $t_{\text{CONV}}$ is the conversion time.
14.4.4 End of conversion sequence (EOS flag)

The ADC notifies the application of each end of sequence (EOS) event. The ADC sets the EOS flag in the ADC_ISR register as soon as the last data result of a conversion sequence is available in the ADC_DR register. An interrupt can be generated if the EOSIE bit is set in the ADC_IER register. The EOS flag is cleared by software by writing 1 to it.

14.4.5 Example timing diagrams (single/continuous modes hardware/software triggers)

Figure 38. Single conversions of a sequence, software trigger

1. EXTEN=00, CONT=0
2. CHSEL=0x20601, WAIT=0, AUTOFF=0
Figure 39. Continuous conversion of a sequence, software trigger

1. EXTEN=00, CONT=1
2. CHSEL=0x20601, WAIT=0, AUTOFF=0

Figure 40. Single conversions of a sequence, hardware trigger

1. EXTSEL=TRGx (over-frequency), EXTEN=01 (rising edge), CONT=0
2. CHSEL=0xF, SCANDIR=0, WAIT=0, AUTOFF=0
**14.4.6 Low frequency trigger mode**

Once the ADC is enabled or the last ADC conversion is complete, the ADC is ready to start a new conversion. The ADC needs to be started at a predefined time \( t_{\text{idle}} \) otherwise ADC converted data might be corrupted due to the transistor leakage (refer to the device datasheet for the maximum value of \( t_{\text{idle}} \)).

If the application has to support a time longer than the maximum \( t_{\text{idle}} \) value (between one trigger to another for single conversion mode or between the ADC enable and the first ADC conversion), then the ADC internal state needs to be rearmed. This mechanism can be enabled by setting LFTRIG bit to 1 in ADC_CFG2 register. By setting this bit, any trigger (software or hardware) will send a rearm command to ADC. The conversion will be started after a two ADC clock cycle delay compared to LFTRIG set to 0.

It is not necessary to use this mode when AUTOFF bit is set to 1. For Wait mode, only the first trigger will generated an internal rearm command.
14.5 Data management

14.5.1 Data register and data alignment (ADC_DR, ALIGN)

At the end of each conversion (when an EOC event occurs), the result of the converted data is stored in the ADC_DR data register which is 16-bit wide.

The format of the ADC_DR depends on the configured data alignment and resolution.

The ALIGN bit in the ADC_CFGR1 register selects the alignment of the data stored after conversion. Data can be right-aligned (ALIGN=0) or left-aligned (ALIGN=1) as shown in Figure 42.

![Figure 42. Data alignment and resolution (oversampling disabled: OVSE = 0)](image)

14.5.2 ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) indicates a data overrun event, when the converted data was not read in time by the CPU or the DMA, before the data from a new conversion is available.

The OVR flag is set in the ADC_ISR register if the EOC flag is still at ‘1’ at the time when a new conversion completes. An interrupt can be generated if the OVRIE bit is set in the ADC_IER register.

When an overrun condition occurs, the ADC keeps operating and can continue to convert unless the software decides to stop and reset the sequence by setting the ADSTP bit in the ADC_CR register.

The OVR flag is cleared by software by writing 1 to it.

It is possible to configure if the data is preserved or overwritten when an overrun event occurs by programming the OVRMOD bit in the ADC_CFGR1 register:

- **OVRMOD=0**
  - An overrun event preserves the data register from being overwritten: the old data is maintained and the new conversion is discarded. If OVR remains at 1, further conversions can be performed but the resulting data is discarded.

- **OVRMOD=1**
  - The data register is overwritten with the last conversion result and the previous unread data is lost. If OVR remains at 1, further conversions can be performed and the ADC_DR register always contains the data from the latest conversion.
14.5.3 Managing a sequence of data converted without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by software. In this case the software must use the EOC flag and its associated interrupt to handle each data result. Each time a conversion is complete, the EOC bit is set in the ADC_ISR register and the ADC_DR register can be read. The OVRMOD bit in the ADC_CFGR1 register should be configured to 0 to manage overrun events as an error.

14.5.4 Managing converted data without using the DMA without overrun

It may be useful to let the ADC convert one or more channels without reading the data after each conversion. In this case, the OVRMOD bit must be configured at 1 and the OVR flag should be ignored by the software. When OVRMOD=1, an overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion data.

14.5.5 Managing converted data using the DMA

Since all converted channel values are stored in a single data register, it is efficient to use DMA when converting more than one channel. This avoids losing the conversion data results stored in the ADC_DR register.

When DMA mode is enabled (DMAEN bit set to 1 in the ADC_CFGR1 register), a DMA request is generated after the conversion of each channel. This allows the transfer of the
converted data from the ADC_DR register to the destination location selected by the software.

**Note:** The DMAEN bit in the ADC_CFGR1 register must be set after the ADC calibration phase.

Despite this, if an overrun occurs (OVR=1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to Section 14.5.2: ADC overrun (OVR, OVRMOD) on page 323).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMACFG in the ADC_CFGR1 register:

- **DMA one shot mode (DMACFG=0)**
  - This mode should be selected when the DMA is programmed to transfer a fixed number of data words.

- **DMA circular mode (DMACFG=1)**
  - This mode should be selected when programming the DMA in circular mode or double buffer mode.

**DMA one shot mode (DMACFG=0)**

In this mode, the ADC generates a DMA transfer request each time a new conversion data word is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when a DMA_EOT interrupt occurs, see Section 9: Direct memory access controller (DMA) on page 241) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted and its partial result discarded
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- The scan sequence is stopped and reset
- The DMA is stopped

**DMA circular mode (DMACFG=1)**

In this mode, the ADC generates a DMA transfer request each time a new conversion data word is available in the data register, even if the DMA has reached the last DMA transfer. This allows the DMA to be configured in circular mode to handle a continuous analog input data stream.
14.6 Low-power features

14.6.1 Wait mode conversion

Wait mode conversion can be used to simplify the software as well as optimizing the performance of applications clocked at low frequency where there might be a risk of ADC overrun occurring.

When the WAIT bit is set to 1 in the ADC_CFGR1 register, a new conversion can start only if the previous data has been treated, once the ADC_DR register has been read or if the EOC bit has been cleared.

This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data.

Note: Any hardware triggers which occur while a conversion is ongoing or during the wait time preceding the read access are ignored.

Figure 44. Wait mode conversion (continuous mode, software trigger)

1. EXTEN=00, CONT=1
2. CHSEL=0x3, SCANDIR=0, WAIT=1, AUTOFF=0
14.6.2 Auto-off mode (AUTOFF)

The ADC has an automatic power management feature which is called auto-off mode, and is enabled by setting AUTOFF=1 in the ADC_CFGR1 register.

When AUTOFF=1, the ADC is always powered off when not converting and automatically wakes-up when a conversion is started (by software or hardware trigger). A startup-time is automatically inserted between the trigger event which starts the conversion and the sampling time of the ADC. The ADC is then automatically disabled once the sequence of conversions is complete.

Auto-off mode can cause a dramatic reduction in the power consumption of applications which need relatively few conversions or when conversion requests are timed far enough apart (for example with a low frequency hardware trigger) to justify the extra power and extra time used for switching the ADC on and off.

Auto-off mode can be combined with the wait mode conversion (WAIT=1) for applications clocked at low frequency. This combination can provide significant power savings if the ADC is automatically powered-off during the wait phase and restarted as soon as the ADC_DR register is read by the application (see Figure 46: Behavior with WAIT=1, AUTOFF=1).

Note: Please refer to the Section Reset and clock control (RCC) for the description of how to manage the dedicated 14 MHz internal oscillator. The ADC interface can automatically switch ON/OFF the 14 MHz internal oscillator to save power.

Figure 45. Behavior with WAIT=0, AUTOFF=1

1. EXTSEL=TRGx, EXTEN=01 (rising edge), CONT=x, ADSTART=1, CHSEL=0xF, SCANDIR=0, WAIT=1, AUTOFF=1
14.7 Analog window watchdog (AWD1EN, AWD1SGL, AWD1CH, ADC_AWDxCR, ADC_AWDxTR)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

14.7.1 Description of analog watchdog 1

AWD1 analog watchdog is enabled by setting the AWD1EN bit in the ADC_CFGR1 register. It is used to monitor that either one selected channel or all enabled channels (see Table 65: Analog watchdog 1 channel selection) remain within a configured voltage range (window) as shown in Figure 47.

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold. These thresholds are programmed in HT1[11:0] and LT1[11:0] bits of ADC_AWD1TR register. An interrupt can be enabled by setting the AWD1IE bit in the ADC_IER register.

The AWD1 flag is cleared by software by programing it to 1.

When converting data with a resolution of less than 12-bit (according to bits DRES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

Table 64 describes how the comparison is performed for all the possible resolutions.
RM0444

Analog-to-digital converter (ADC)

### Table 64. Analog watchdog comparison

<table>
<thead>
<tr>
<th>Resolution bits RES[1:0]</th>
<th>Analog Watchdog comparison between:</th>
<th>Thresholds</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Raw converted data, left aligned(^{(1)})</td>
<td>LTx[11:0] and HTx[11:0]</td>
<td></td>
</tr>
<tr>
<td>00: 12-bit</td>
<td>DATA[11:0]</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>01: 10-bit</td>
<td>DATA[11:2],00</td>
<td>LTx[11:0] and HTx[11:0]</td>
<td>The user must configure LTx[1:0] and HTx[1:0] to &quot;00&quot;</td>
</tr>
<tr>
<td>10: 8-bit</td>
<td>DATA[11:4],0000</td>
<td>LTx[11:0] and HTx[11:0]</td>
<td>The user must configure LTx[3:0] and HTx[3:0] to &quot;0000&quot;</td>
</tr>
</tbody>
</table>

\(^{(1)}\) The watchdog comparison is performed on the raw converted data before any alignment calculation.

Table 65 shows how to configure the AWD1SGL and AWD1EN bits in the ADC_CFGR1 register to enable the analog watchdog on one or more channels.

### Figure 47. Analog watchdog guarded area

![Analog voltage guarded area diagram](MS45396V1)

### Table 65. Analog watchdog 1 channel selection

<table>
<thead>
<tr>
<th>Channels guarded by the analog watchdog</th>
<th>AWD1SGL bit</th>
<th>AWD1EN bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>All channels</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Single(^{(1)}) channel</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\(^{(1)}\) Selected by the AWD1CH[4:0] bits

### 14.7.2 Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the AWDxCHy in ADC_AWDxCR (x = 2, 3).

The corresponding watchdog is enabled when any AWDxCHy bit (x=2,3) is set in ADC_AWDxCR register.

When converting data with a resolution of less than 12 bits (configured through DRES[1:0] bits), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

Table 64 describes how the comparison is performed for all the possible resolutions.

The AWD2/3 analog watchdog status bit is set if the analog voltage converted by the ADC is below a low threshold or above a high threshold. These thresholds are programmed in
HTx[11:0] and LTx[11:0] of ADC_AWDxTR registers (x=2 or 3). An interrupt can be enabled by setting the AWDxIE bit in the ADC_IER register.

The AWD2 and ADW3 flags are cleared by software by programming them to 1.

### 14.7.3 ADC_AWDx_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal, ADC_AWDx_OUT (x being the watchdog number) that is directly connected to the ETR input (external trigger) of some on-chip timers (refer to the timers section for details on how to select the ADC_AWDx_OUT signal as ETR).

ADC_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADC_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC_AWDx_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds. It remains at 1 if the next guarded conversions are still outside the programmed thresholds.
- ADC_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS to 1). Note that stopping conversions (ADSTP set to 1), might clear the ADC_AWDx_OUT state.
- ADC_AWDx_OUT state does not change when the ADC converts the none-guarded channel (see Figure 50).

AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADC_AWDx_OUT (as an example, ADC_AWDx_OUT can toggle while AWDx flag remains at 1 if the software has not cleared the flag).

The ADC_AWDx_OUT signal is generated by the ADC_CLK domain. This signal can be generated even the APB clock is stopped.

The AWD comparison is performed at the end of each ADC conversion. The ADC_AWDx_OUT rising edge and falling edge occurs two ADC_CLK clock cycles after the comparison.

As ADC_AWDx_OUT is generated by the ADC_CLK domain and AWD flag is generated by the APB clock domain, the rising edges of these signals are not synchronized.

**Figure 48. ADC_AWDx_OUT signal generation**

<table>
<thead>
<tr>
<th>ADC STATE</th>
<th>Conversion1 inside</th>
<th>Conversion2 outside</th>
<th>Conversion3 inside</th>
<th>Conversion4 outside</th>
<th>Conversion5 outside</th>
<th>Conversion6 outside</th>
<th>Conversion7 inside</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOC FLAG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWDx FLAG</td>
<td>Cleared by SW</td>
<td>Cleared by SW</td>
<td>Cleared by SW</td>
<td>Cleared by SW</td>
<td>Cleared by SW</td>
<td>Cleared by SW</td>
<td></td>
</tr>
<tr>
<td>ADC_AWDx_OUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Converted channels: 1,2,3,4,5,6,7
Guarded converted channels: 1,2,3,4,5,6,7
14.7.4 **Analog Watchdog threshold control**

LTx[11:0] and HTx[11:0] can be changed during an analog-to-digital conversion (that is between the start of the conversion and the end of conversion of the ADC internal state). If HTx and LTx bits are programmed during the ADC guarded channel conversion, the watchdog function is masked for this conversion. This mask is cleared when starting a new conversion, and the resulting new AWD threshold will be applied starting the next ADC conversion result. AWD comparison is performed at each end of conversion. If the current ADC data are out of the new threshold interval, this does not generated any interrupt or an ADC_AWDx_OUT signal. The Interrupt and the ADC_AWDx_OUT generation only occurs at the end of the ADC conversion that started after the threshold update. If ADC_AWDx_OUT is already asserted, programming the new threshold does not deassert the ADC_AWDx_OUT signal.
14.8 Oversampler

The oversampling unit performs data preprocessing to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 16-bit.

It provides a result with the following form, where N and M can be adjusted:

\[ \text{Result} = \frac{1}{M} \sum_{n=0}^{N-1} \text{Conversion}(t_n) \]

It allows to perform by hardware the following functions: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVFS[2:0] bits in the ADC_CFRG2 register. It can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits. It is configured through the OVSS[3:0] bits in the ADC_CFRG2 register.

The summation unit can yield a result up to 20 bits (256 x 12-bit), which is first shifted right. The upper bits of the result are then truncated, keeping only the 16 least significant bits rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the ADC_DR data register.

**Note:** If the intermediate result after the shifting exceeds 16 bits, the upper bits of the result are simply truncated.
The Figure 53 gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

![Figure 53. Numerical example with 5-bits shift and rounding](image)

The Table 66 below gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFF.

<table>
<thead>
<tr>
<th>Oversampling ratio</th>
<th>Max Raw data</th>
<th>No-shift OVSS = 0000</th>
<th>1-bit shift OVSS = 0001</th>
<th>2-bit shift OVSS = 0010</th>
<th>3-bit shift OVSS = 0111</th>
<th>4-bit shift OVSS = 0100</th>
<th>5-bit shift OVSS = 0101</th>
<th>6-bit shift OVSS = 0110</th>
<th>7-bit shift OVSS = 0111</th>
<th>8-bit shift OVSS = 1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2x</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
</tr>
<tr>
<td>4x</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
</tr>
<tr>
<td>8x</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
</tr>
<tr>
<td>16x</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
</tr>
<tr>
<td>32x</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
</tr>
<tr>
<td>64x</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
</tr>
<tr>
<td>128x</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
</tr>
<tr>
<td>256x</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
<td>0xFFF00</td>
</tr>
</tbody>
</table>

The conversion timings in oversampled mode do not change compared to standard conversion mode: the sample time is maintained equal during the whole oversampling sequence. New data are provided every N conversion, with an equivalent delay equal to N x tCONV = N x (tSMPL + tSAR). The flags features are raised as following:

- the end of the sampling phase (EOSMP) is set after each sampling phase
- the end of conversion (EOC) occurs once every N conversions, when the oversampled result is available
- the end of sequence (EOCSEQ) occurs once the sequence of oversampled data is completed (i.e. after N x sequence length conversions total)
14.8.1 ADC operating modes supported when oversampling

In oversampling mode, most of the ADC operating modes are available:

- Single or continuous mode conversions, forward or backward scanned sequences and up to 8 channels programmed sequence
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (WAIT, AUTOFF)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC_CFGR1 register) are accumulated, truncated, rounded and shifted in the same way as 12-bit conversions are

Note: The alignment mode is not available when working with oversampled data. The ALIGN bit in ADC_CFGR1 is ignored and the data are always provided right-aligned.

14.8.2 Analog watchdog

The analog watchdog functionality is available (AWDxSGL, AW1DEN and AWDxCH bits), with the following differences:

- the RES[1:0] bits are ignored, comparison is always done on using the full 12-bits values HTx[11:0] and LTx[11:0]
- the comparison is performed on the most significant 12 bits of the 16 bits oversampled results ADC_DR[15:4]

Note: Care must be taken when using high shifting values. This reduces the comparison range. For instance, if the oversampled result is shifted by 4 bits thus yielding a 12-bit data right-aligned, the effective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HTx[7:0] / LTx[7:0], and HTx[11:8] / LTx[11:8] must be kept reset.

14.8.3 Triggered mode

The averager can also be used for basic filtering purposes. Although not a very efficient filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TOVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

Figure 54 below shows how conversions are started in response to triggers in discontinuous mode.

If the TOVS bit is set, the content of the DISCEN bit is ignored and considered as 1.
14.9 Temperature sensor and internal reference voltage

The temperature sensor can be used to measure the junction temperature ($T_J$) of the device. The temperature sensor is internally connected to the ADC $V_{IN}[12]$ input channel which is used to convert the sensor's output voltage to a digital value. The sampling time for the temperature sensor analog pin must be greater than the minimum $T_{S_{\text{temp}}}$ value specified in the datasheet. When not in use, the sensor can be put in power down mode.

The internal voltage reference (VREFINT) provides a stable (bandgap) voltage output for the ADC and comparators. VREFINT is internally connected to the ADC $V_{IN}[13]$ input channel. The precise voltage of VREFINT is individually measured for each part by ST during production test and stored in the system memory area.

Figure 55 shows the block diagram of connections between the temperature sensor, the internal voltage reference and the ADC.

The TSEN bit must be set to enable the conversion of ADC $V_{IN}[12]$ (temperature sensor) and the VREFEN bit must be set to enable the conversion of ADC $V_{IN}[13]$ (VREFINT).

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference. Refer to the datasheet for additional information.
Main features

- Supported temperature range: −40 to 125 °C
- Linearity: ±2 °C max., precision depending on calibration

**Figure 55. Temperature sensor and VREFINT channel block diagram**

Reading the temperature

1. Select the ADC V_IN[12] input channel
2. Select an appropriate sampling time specified in the device datasheet (T_S_temp).
3. Set the TSEN bit in the ADC_CCR register to wake up the temperature sensor from power down mode and wait for its stabilization time (t_START).
4. Start the ADC conversion by setting the ADSTART bit in the ADC_CR register (or by external trigger)
5. Read the resulting V_TS data in the ADC_DR register
6. Calculate the temperature using the following formula

\[
\text{Temperature (in °C)} = \frac{\text{TS_CAL2_TEMP} - \text{TS_CAL1_TEMP}}{\text{TS_CAL2} - \text{TS_CAL1}} \times (\text{TS_DATA} - \text{TS_CAL1}) + \text{TS_CAL1_TEMP}
\]

Where:

- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP (refer to the datasheet for TS_CAL2 value)
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP (refer to the datasheet for TS_CAL1 value)
- TS_DATA is the actual temperature sensor output value converted by ADC

Refer to the specific device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

**Note:** The sensor has a startup time after waking from power down mode before it can output V_TS at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and TSEN bits should be set at the same time.
Calculating the actual $V_{DDA}$ voltage using the internal reference voltage

The $V_{DDA}$ power supply voltage applied to the microcontroller may be subject to variation or not precisely known. The embedded internal voltage reference ($V_{REFINT}$) and its calibration data acquired by the ADC during the manufacturing process at $V_{DDA} = 3.3$ V can be used to evaluate the actual $V_{DDA}$ voltage level.

The following formula gives the actual $V_{DDA}$ voltage supplying the device:

$$V_{DDA} = 3 \times \frac{V_{REFINT\_CAL}}{V_{REFINT\_DATA}}$$

Where:
- $V_{REFINT\_CAL}$ is the $V_{REFINT}$ calibration value
- $V_{REFINT\_DATA}$ is the actual $V_{REFINT}$ output value converted by ADC

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between the analog power supply and the voltage applied on the converted channel. For most application use cases, it is necessary to convert this ratio into a voltage independent of $V_{DDA}$. For applications where $V_{DDA}$ is known and ADC converted values are right-aligned, you can use the following formula to get this absolute value:

$$V_{CHANNEL_x} = \frac{3 \times V_{REFINT\_CAL} \times ADC\_DATA_x}{V_{REFINT\_DATA} \times \text{FULL}\_\text{SCALE}}$$

For applications where $V_{DDA}$ value is not known, you must use the internal voltage reference and $V_{DDA}$ can be replaced by the expression provided in the section Calculating the actual $V_{DDA}$ voltage using the internal reference voltage, resulting in the following formula:

$$V_{CHANNEL_x} = \frac{3 \times V_{REFINT\_CAL} \times ADC\_DATA_x}{V_{REFINT\_DATA} \times \text{FULL}\_\text{SCALE}}$$

Where:
- $V_{REFINT\_CAL}$ is the $V_{REFINT}$ calibration value
- $ADC\_DATA_x$ is the value measured by the ADC on channelx (right-aligned)
- $V_{REFINT\_DATA}$ is the actual $V_{REFINT}$ output value converted by the ADC
- $\text{full}\_\text{SCALE}$ is the maximum digital value of the ADC output. For example with 12-bit resolution, it will be $2^{12} - 1 = 4095$ or with 8-bit resolution, $2^8 - 1 = 255$.

Note: If ADC measurements are done using an output format other than 12 bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

14.10 Battery voltage monitoring

The $V_{BATEN}$ bit in the ADC_CCR register allows the application to measure the backup battery voltage on the $V_{BAT}$ pin. As the $V_{BAT}$ voltage could be higher than $V_{DDA}$, to ensure the correct operation of the ADC, the $V_{BAT}$ pin is internally connected to a bridge divider. This bridge is automatically enabled when $V_{BATEN}$ is set, to connect $V_{BAT}$ to the ADC $V_{IN[14]}$ input channel. As a consequence, the converted digital value is half the $V_{BAT}$ voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed for ADC conversion.
14.11 ADC interrupts

An interrupt can be generated by any of the following events:

- End Of Calibration (EOCAL flag)
- ADC power-up, when the ADC is ready (ADRDY flag)
- End of any conversion (EOC flag)
- End of a sequence of conversions (EOS flag)
- When an analog watchdog detection occurs (AWD1, AWD2, AWD3 flags)
- When the Channel configuration is ready (CCRDY flag)
- When the end of sampling phase occurs (EOSMP flag)
- when a data overrun occurs (OVR flag)

Separate interrupt enable bits are available for flexibility.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>End Of Calibration</td>
<td>EOCAL</td>
<td>EOCALIE</td>
</tr>
<tr>
<td>ADC ready</td>
<td>ADRDY</td>
<td>ADRDYIE</td>
</tr>
<tr>
<td>End of conversion</td>
<td>EOC</td>
<td>EOCIE</td>
</tr>
<tr>
<td>End of sequence of conversions</td>
<td>EOS</td>
<td>EOSIE</td>
</tr>
<tr>
<td>Analog watchdog 1 status bit is set</td>
<td>AWD1</td>
<td>AWD1IE</td>
</tr>
<tr>
<td>Analog watchdog 2 status bit is set</td>
<td>AWD2</td>
<td>AWD2IE</td>
</tr>
<tr>
<td>Analog watchdog 3 status bit is set</td>
<td>AWD3</td>
<td>AWD3IE</td>
</tr>
<tr>
<td>Channel Configuration Ready</td>
<td>CCRDY</td>
<td>CCRDYIE</td>
</tr>
</tbody>
</table>
### Table 67. ADC interrupts (continued)

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>End of sampling phase</td>
<td>EOSMP</td>
<td>EOSMPIE</td>
</tr>
<tr>
<td>Overrun</td>
<td>OVR</td>
<td>OVRIE</td>
</tr>
</tbody>
</table>
14.12 ADC registers

Refer to Section 1.2 on page 50 for a list of abbreviations used in register descriptions.

14.12.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x00
Reset value: 0x0000 0000

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **CCRDY**: Channel Configuration Ready flag

This flag bit is set by hardware when the channel configuration is applied after programming to ADC_CHSELR register or changing CHSELRMOD or SCANDIR. It is cleared by software by programming it to 1.

0: Channel configuration update not applied.
1: Channel configuration update is applied.

**Note**: When the software configures the channels (by programming ADC_CHSELR or changing CHSELRMOD or SCANDIR), it must wait until the CCRDY flag rises before configuring again or starting conversions, otherwise the new configuration (or the START bit) will be ignored. Once the flag is asserted, if the software needs to configure again the channels, it must clear the CCRDY flag before proceeding with a new configuration.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **EOCAL**: End Of Calibration flag

This bit is set by hardware when calibration is complete. It is cleared by software writing 1 to it.

0: Calibration is not complete
1: Calibration is complete

Bit 10 Reserved, must be kept at reset value.

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in ADC_AWD3TR and ADC_AWD3TR registers. It is cleared by software by programming it to 1.

0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)
1: Analog watchdog event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in ADC_AWD2TR and ADC_AWD2TR registers. It is cleared by software programming it to 1.

0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)
1: Analog watchdog event occurred
Bit 7  **AWD1**: Analog watchdog 1 flag
This bit is set by hardware when the converted voltage crosses the values programmed in ADC_TR1 and ADC_HR1 registers. It is cleared by software by programming it to 1.
0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)
1: Analog watchdog event occurred

Bits 6:5 Reserved, must be kept at reset value.

Bit 4  **OVR**: ADC overrun
This bit is set by hardware when an overrun occurs, meaning that a new conversion has complete while the EOC flag was already set. It is cleared by software writing 1 to it.
0: No overrun occurred (or the flag event was already acknowledged and cleared by software)
1: Overrun has occurred

Bit 3  **EOS**: End of sequence flag
This bit is set by hardware at the end of the conversion of a sequence of channels selected by the CHSEL bits. It is cleared by software writing 1 to it.
0: Conversion sequence not complete (or the flag event was already acknowledged and cleared by software)
1: Conversion sequence complete

Bit 2  **EOC**: End of conversion flag
This bit is set by hardware at the end of each conversion of a channel when a new data result is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register.
0: Channel conversion not complete (or the flag event was already acknowledged and cleared by software)
1: Channel conversion complete

Bit 1  **EOSMP**: End of sampling flag
This bit is set by hardware during the conversion, at the end of the sampling phase. It is cleared by software by programming it to ‘1’.
0: Not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)
1: End of sampling phase reached

Bit 0  **ADRDY**: ADC ready
This bit is set by hardware after the ADC has been enabled (bit ADEN=1) and when the ADC reaches a state where it is ready to accept conversion requests. It is cleared by software writing 1 to it.
0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)
1: ADC is ready to start conversion
14.12.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x04
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>Res.</td>
<td>CCRDYIE</td>
<td>Res.</td>
<td>EOCALIE</td>
<td>Res.</td>
<td>AWD3IE</td>
<td>Res.</td>
<td>AWD2IE</td>
<td>Res.</td>
<td>AWD1IE</td>
<td>Res.</td>
<td>OVRIE</td>
<td>EOSIE</td>
<td>EOCIE</td>
<td>EOSMPIE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 CCRDYIE: Channel Configuration Ready Interrupt enable
This bit is set and cleared by software to enable/disable the channel configuration ready interrupt.
0: Channel configuration ready interrupt disabled
1: Channel configuration ready interrupt enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 12 Reserved, must be kept at reset value.

Bit 11 EOCALIE: End of calibration interrupt enable
This bit is set and cleared by software to enable/disable the end of calibration interrupt.
0: End of calibration interrupt disabled
1: End of calibration interrupt enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 10 Reserved, must be kept at reset value.

Bit 9 AWD3IE: Analog watchdog 3 interrupt enable
This bit is set and cleared by software to enable/disable the analog watchdog interrupt.
0: Analog watchdog interrupt disabled
1: Analog watchdog interrupt enabled

Note: The Software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 8 AWD2IE: Analog watchdog 2 interrupt enable
This bit is set and cleared by software to enable/disable the analog watchdog interrupt.
0: Analog watchdog interrupt disabled
1: Analog watchdog interrupt enabled

Note: The Software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 7 AWD1IE: Analog watchdog 1 interrupt enable
This bit is set and cleared by software to enable/disable the analog watchdog interrupt.
0: Analog watchdog interrupt disabled
1: Analog watchdog interrupt enabled

Note: The Software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bits 6:5 Reserved, must be kept at reset value.
Bit 4 OVRIE: Overrun interrupt enable
This bit is set and cleared by software to enable/disable the overrun interrupt.
0: Overrun interrupt disabled
1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.
Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 3 EOSIE: End of conversion sequence interrupt enable
This bit is set and cleared by software to enable/disable the end of sequence of conversions interrupt.
0: EOS interrupt disabled
1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.
Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 2 EOCIE: End of conversion interrupt enable
This bit is set and cleared by software to enable/disable the end of conversion interrupt.
0: EOC interrupt disabled
1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.
Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 1 EOSMPIE: End of sampling flag interrupt enable
This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt.
0: EOSMP interrupt disabled.
1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.
Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 0 ADRDYIE: ADC ready interrupt enable
This bit is set and cleared by software to enable/disable the ADC Ready interrupt.
0: ADRDY interrupt disabled.
1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.
Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).
14.12.3 ADC control register (ADC_CR)

Address offset: 0x08
Reset value: 0x0000 0000

### Bit 31 ADCAL: ADC calibration
This bit is set by software to start the calibration of the ADC. It is cleared by hardware after calibration is complete.
- 0: Calibration complete
- 1: Write 1 to calibrate the ADC. Read at 1 means that a calibration is in progress.

**Note:** The software is allowed to set ADCAL only when the ADC is disabled (ADCAL=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0). The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN=1 and ADSTART=0 (ADC enabled and no conversion is ongoing).

### Bits 30:29 Reserved, must be kept at reset value.

### Bit 28 ADVREGEN: ADC Voltage Regulator Enable
This bit is set by software to enable the ADC internal voltage regulator. The voltage regulator output is available after tADCVREG_SETUP.
- 0: ADC voltage regulator disabled
- 1: ADC voltage regulator enabled

**Note:** The software can program this bit field only when the ADC is disabled (ADCAL=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

### Bits 27:5 Reserved, must be kept at reset value.

### Bit 4 ADSTP: ADC stop conversion command
This bit is set by software to stop and discard an ongoing conversion (ADSTP Command).
- 0: No ADC stop conversion command ongoing
- 1: Write 1 to stop the ADC. Read 1 means that an ADSTP command is in progress.

**Note:** Setting ADSTP to ‘1’ is only effective when ADSTART=1 and ADDIS=0 (ADC is enabled and may be converting and there is no pending request to disable the ADC)

### Bit 3 Reserved, must be kept at reset value.
Bit 2  **ADSTART**: ADC start conversion command

This bit is set by software to start ADC conversion. Depending on the EXTEN [1:0] configuration bits, a conversion either starts immediately (software trigger configuration) or once a hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- In single conversion mode (CONT=0, DISCEN=0), when software trigger is selected (EXTEN=00): at the assertion of the end of Conversion Sequence (EOS) flag.
- In discontinuous conversion mode (CONT=0, DISCEN=1), when the software trigger is selected (EXTEN=00): at the assertion of the end of Conversion (EOC) flag.
- In all other cases: after the execution of the ADSTP command, at the same time as the ADSTP bit is cleared by hardware.

| 0 | No ADC conversion is ongoing. |
| 1 | Write 1 to start the ADC. Read 1 means that the ADC is operating and may be converting. |

Note: The software is allowed to set ADSTART only when ADEN=1 and ADDIS=0 (ADC is enabled and there is no pending request to disable the ADC).

After writing to ADC_CHSELR register or changing CHSELMOD or SCANDIRW, it is mandatory to wait until CCRDY flag is asserted before setting ADSTART, otherwise, the value written to ADSTART will be ignored.

Bit 1  **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

| 0 | No ADDIS command ongoing |
| 1 | Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress. |

Note: Setting ADDIS to ‘1’ is only effective when ADEN=1 and ADSTART=0 (which ensures that no conversion is ongoing).

Bit 0  **ADEN**: ADC enable command

This bit is set by software to enable the ADC. The ADC will be effectively ready to operate once the ADRDY flag has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

| 0 | ADC is disabled (OFF state) |
| 1 | Write 1 to enable the ADC. |

Note: The software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADCAL=0, ADSTP=0, ADSTART=0, ADDIS=0 and ADEN=0)
14.12.4 ADC configuration register 1 (ADC_CFGR1)

Address offset: 0x0C
Reset value: 0x0000 0000

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

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **AWD1CH[4:0]:** Analog watchdog channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

- 00000: ADC analog input Channel 0 monitored by AWD
- 00001: ADC analog input Channel 1 monitored by AWD
- ....
- 10001: ADC analog input Channel 17 monitored by AWD
- 10010: ADC analog input Channel 18 monitored by AWD
- Others: Reserved

*Note:* The channel selected by the AWDCH[4:0] bits must be also set into the CHSELR register.

*The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).*

Bits 25:24 Reserved, must be kept at reset value.

Bit 23 **AWD1EN:** Analog watchdog enable

This bit is set and cleared by software.

- 0: Analog watchdog 1 disabled
- 1: Analog watchdog 1 enabled

*Note:* The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 22 **AWD1SG:** Enable the watchdog on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWDCH[4:0] bits or on all the channels

- 0: Analog watchdog 1 enabled on all channels
- 1: Analog watchdog 1 enabled on a single channel

*Note:* The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).
Bit 21 **CHSELRMOD**: Mode selection of the ADC_CHSELR register

This bit is set and cleared by software to control the ADC_CHSELR feature:

0: Each bit of the ADC_CHSELR register enables an input
1: ADC_CHSELR register is able to sequence up to 8 channels

*Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).*

If CCRDY is not yet asserted after channel configuration (writing ADC_CHSELR register or changing CHSELRMOD or SCANDIR), the value written to this bit will be ignored.

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **DISCEN**: Discontinuous mode

This bit is set and cleared by software to enable/disable discontinuous mode.

0: Discontinuous mode disabled
1: Discontinuous mode enabled

*Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN=1 and CONT=1. The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).*

Bit 15 **AUTOFF**: Auto-off mode

This bit is set and cleared by software to enable/disable auto-off mode.

0: Auto-off mode disabled
1: Auto-off mode enabled

*Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).*

Bit 14 **WAIT**: Wait conversion mode

This bit is set and cleared by software to enable/disable wait conversion mode.

0: Wait conversion mode off
1: Wait conversion mode on

*Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).*

Bit 13 **CONT**: Single / continuous conversion mode

This bit is set and cleared by software. If it is set, conversion takes place continuously until it is cleared.

0: Single conversion mode
1: Continuous conversion mode

*Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN=1 and CONT=1. The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).*

Bit 12 **OVRMOD**: Overrun management mode

This bit is set and cleared by software and configure the way data overruns are managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.
1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

*Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).*
Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection
These bits are set and cleared by software to select the external trigger polarity and enable the trigger.
00: Hardware trigger detection disabled (conversions can be started by software)
01: Hardware trigger detection on the rising edge
10: Hardware trigger detection on the falling edge
11: Hardware trigger detection on both the rising and falling edges

*Note:* The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 9 Reserved, must be kept at reset value.

Bits 8:6 **EXTSEL[2:0]**: External trigger selection
These bits select the external event used to trigger the start of conversion (refer to Table 62: External triggers for details):
000: TRG0
001: TRG1
010: TRG2
011: TRG3
100: TRG4
101: TRG5
110: TRG6
111: TRG7

*Note:* The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 5 **ALIGN**: Data alignment
This bit is set and cleared by software to select right or left alignment. Refer to Figure 42: Data alignment and resolution (oversampling disabled: OVSE = 0) on page 323
0: Right alignment
1: Left alignment

*Note:* The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bits 4:3 **RES[1:0]**: Data resolution
These bits are written by software to select the resolution of the conversion.
00: 12 bits
01: 10 bits
10: 8 bits
11: 6 bits

*Note:* The software is allowed to write these bits only when ADEN=0.
Bit 2 **SCANDIR**: Scan sequence direction

This bit is set and cleared by software to select the direction in which the channels will be scanned in the sequence. It is effective only if CHSELMOD bit is cleared to 0.

0: Upward scan (from CHSEL0 to CHSEL18)
1: Backward scan (from CHSEL18 to CHSEL0)

*Note*: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

If CCRDY is not yet asserted after channel configuration (writing ADC_CHSELR register or changing CHSELMOD or SCANDIR), the value written to this bit will be ignored.

Bit 1 **DMACFG**: Direct memory access configuration

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN=1.

0: DMA one shot mode selected
1: DMA circular mode selected

For more details, refer to Section 14.5.5: Managing converted data using the DMA on page 324

*Note*: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 0 **DMAEN**: Direct memory access enable

This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the DMA controller to manage automatically the converted data. For more details, refer to Section 14.5.5: Managing converted data using the DMA on page 324.

0: DMA disabled
1: DMA enabled

*Note*: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).
14.12.5  ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-30</th>
<th>CKMODE[1:0]: ADC clock mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>These bits are set and cleared by software to define how the analog ADC is clocked:</td>
</tr>
<tr>
<td></td>
<td>00: ADCCLK (Asynchronous clock mode), generated at product level (refer to RCC section)</td>
</tr>
<tr>
<td></td>
<td>01: PCLK/2 (Synchronous clock mode)</td>
</tr>
<tr>
<td></td>
<td>10: PCLK/4 (Synchronous clock mode)</td>
</tr>
<tr>
<td></td>
<td>11: PCLK (Synchronous clock mode). This configuration must be enabled only if PCLK has a 50% duty clock cycle (APB prescaler configured inside the RCC must be bypassed and the system clock must by 50% duty cycle)</td>
</tr>
<tr>
<td></td>
<td>In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.</td>
</tr>
<tr>
<td></td>
<td>Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>LFTRIG: Low frequency trigger mode enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: Low Frequency Trigger Mode disabled</td>
</tr>
<tr>
<td></td>
<td>1: Low Frequency Trigger Mode enabled</td>
</tr>
<tr>
<td></td>
<td>Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 (this ensures that no conversion is ongoing).</td>
</tr>
</tbody>
</table>

| Bit 28-10 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bit 9</th>
<th>TOVS: Triggered Oversampling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: All oversampled conversions for a channel are done consecutively after a trigger</td>
</tr>
<tr>
<td></td>
<td>1: Each oversampled conversion for a channel needs a trigger</td>
</tr>
<tr>
<td></td>
<td>Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).</td>
</tr>
</tbody>
</table>
Bits 8:5 **OVSS[3:0]: Oversampling shift**
This bit is set and cleared by software.
- 0000: No shift
- 0001: Shift 1-bit
- 0010: Shift 2-bits
- 0011: Shift 3-bits
- 0100: Shift 4-bits
- 0101: Shift 5-bits
- 0110: Shift 6-bits
- 0111: Shift 7-bits
- 1000: Shift 8-bits
- Others: Reserved

*Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).*

Bits 4:2 **OVSR[2:0]: Oversampling ratio**
This bit filed defines the number of oversampling ratio.
- 000: 2x
- 001: 4x
- 010: 8x
- 011: 16x
- 100: 32x
- 101: 64x
- 110: 128x
- 111: 256x

*Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).*

Bit 1 Reserved, must be kept at reset value.

Bit 0 **OVSE: Oversampler Enable**
This bit is set and cleared by software.
- 0: Oversampler disabled
- 1: Oversampler enabled

*Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).*

### 14.12.6 ADC sampling time register (ADC_SMPR)

**Address offset:** 0x14

**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMPSE L18</td>
<td>SMPSE L17</td>
<td>SMPSE L16</td>
<td>SMPSE L15</td>
<td>SMPSE L14</td>
<td>SMPSE L13</td>
<td>SMPSE L12</td>
<td>SMPSE L11</td>
<td>SMPSE L10</td>
<td>SMPSE L9</td>
<td>SMPSE L8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SMPSE L7</td>
<td>SMPSE L6</td>
<td>SMPSE L5</td>
<td>SMPSE L4</td>
<td>SMPSE L3</td>
<td>SMPSE L2</td>
<td>SMPSE L1</td>
<td>SMPSE L0</td>
<td>Res.</td>
<td>Res.</td>
<td>SMP[2:0]</td>
<td>Res.</td>
<td>SMP[2:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:27 Reserved, must be kept at reset value.
Bits 26:8 **SMPSEL[18:0]** Channel-x sampling time selection

These bits are written by software to define which sampling time will be used.
0: Sampling time of CHANNELx use the setting of SMP1[2:0] register.
1: Sampling time of CHANNELx use the setting of SMP2[2:0] register.

*Note:* The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SMP2[2:0]**: Sampling time selection 2

These bits are written by software to select the sampling time that applies to all channels.
000: 1.5 ADC clock cycles
001: 3.5 ADC clock cycles
010: 7.5 ADC clock cycles
011: 12.5 ADC clock cycles
100: 19.5 ADC clock cycles
101: 39.5 ADC clock cycles
110: 79.5 ADC clock cycles
111: 160.5 ADC clock cycles

*Note:* The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMP1[2:0]**: Sampling time selection 1

These bits are written by software to select the sampling time that applies to all channels.
000: 1.5 ADC clock cycles
001: 3.5 ADC clock cycles
010: 7.5 ADC clock cycles
011: 12.5 ADC clock cycles
100: 19.5 ADC clock cycles
101: 39.5 ADC clock cycles
110: 79.5 ADC clock cycles
111: 160.5 ADC clock cycles

*Note:* The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

---

**14.12.7 ADC watchdog threshold register (ADC_AWD1TR)**

Address offset: 0x20

Reset value: 0x0FFF 0000

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

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

Bits 31:28 Reserved, must be kept at reset value.
Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog.
Refer to Section 14.7: Analog window watchdog (AWD1EN, AWD1SGL, AWD1CH, ADC_AWDxCR, ADC_AWDxTR) on page 328.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **LT1[11:0]**: Analog watchdog 1 lower threshold
These bits are written by software to define the lower threshold for the analog watchdog.
Refer to Section 14.7: Analog window watchdog (AWD1EN, AWD1SGL, AWD1CH, ADC_AWDxCR, ADC_AWDxTR) on page 328.

### 14.12.8 ADC watchdog threshold register (ADC_AWD2TR)

Address offset: 0x24
Reset value: 0x0FFF 0000

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Description</th>
<th>Read/Write</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-28</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw, rw, rw, rw</td>
<td></td>
</tr>
<tr>
<td>27-16</td>
<td><strong>HT2[11:0]</strong>: Analog watchdog 2 higher threshold</td>
<td>rw, rw, rw, rw</td>
<td></td>
</tr>
<tr>
<td>15-12</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw, rw, rw, rw</td>
<td></td>
</tr>
<tr>
<td>11-0</td>
<td><strong>LT2[11:0]</strong>: Analog watchdog 2 lower threshold</td>
<td>rw, rw, rw, rw</td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 27:16 **HT2[11:0]**: Analog watchdog 2 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog.
Refer to Section 14.7: Analog window watchdog (AWD1EN, AWD1SGL, AWD1CH, ADC_AWDxCR, ADC_AWDxTR) on page 328.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **LT2[11:0]**: Analog watchdog 2 lower threshold
These bits are written by software to define the lower threshold for the analog watchdog.
Refer to Section 14.7: Analog window watchdog (AWD1EN, AWD1SGL, AWD1CH, ADC_AWDxCR, ADC_AWDxTR) on page 328.
14.12.9 ADC channel selection register [alternate] (ADC_CHSELR)

Address offset: 0x28
Reset value: 0x0000 0000

The same register can be used in two different modes:
– Each ADC_CHSELR bit enables an input (CHSELRMOD = 0 in ADC_CFG1). Refer to the current section.
– ADC_CHSELR is able to sequence up to 8 channels (CHSELRMOD = 1 in ADC_CFG1). Refer to next section.

CHSELRMOD = 0 in ADC_CFG1:

<table>
<thead>
<tr>
<th>Bits (31:19)</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 18:0</td>
<td>CHSEL[18:0]: Channel-x selection</td>
</tr>
<tr>
<td></td>
<td>These bits are written by software and define which channels are part of the sequence of channels to be converted.</td>
</tr>
<tr>
<td></td>
<td>0: Input Channel-x is not selected for conversion</td>
</tr>
<tr>
<td></td>
<td>1: Input Channel-x is selected for conversion</td>
</tr>
<tr>
<td>Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).</td>
<td></td>
</tr>
<tr>
<td>If CCRDY is not yet asserted after channel configuration (writing ADC_CHSELR register or changing CHSELRMOD or SCANDIR), the value written to this bit will be ignored.</td>
<td></td>
</tr>
</tbody>
</table>
14.12.10 ADC channel selection register [alternate] (ADC_CHSELR)

Address offset: 0x28
Reset value: 0x0000 0000

The same register can be used in two different modes:
- Each ADC_CHSELR bit enables an input (CHSELRMOD = 0 in ADC_CFGR1). Refer to the current previous section.
- ADC_CHSELR is able to sequence up to 8 channels (CHSELRMOD = 1 in ADC_CFGR1). Refer to this section.

CHSELRMOD = 1 in ADC_CFGR1:

<table>
<thead>
<tr>
<th>CHSELRMOD = 1 in ADC_CFGR1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>Bits 31:28 SQ8[3:0]: 8th conversion of the sequence</td>
</tr>
</tbody>
</table>
These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates the end of the sequence.
When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.
0000: CH0
0001: CH1
... 1100: CH12
1101: CH13
1110: CH14
1111: No channel selected (End of sequence)

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 27:24 SQ7[3:0]: 7th conversion of the sequence
These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.
When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.
Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 23:20 SQ6[3:0]: 6th conversion of the sequence
These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.
When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.
Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).
Bits 19:16 **SQ5[3:0]**: 5th conversion of the sequence  
These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.  
When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.  
Refer to SQ8[3:0] for a definition of channel selection.  
*Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).*

Bits 15:12 **SQ4[3:0]**: 4th conversion of the sequence  
These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.  
When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.  
Refer to SQ8[3:0] for a definition of channel selection.  
*Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).*

Bits 11:8 **SQ3[3:0]**: 3rd conversion of the sequence  
These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.  
When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.  
Refer to SQ8[3:0] for a definition of channel selection.  
*Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).*

Bits 7:4 **SQ2[3:0]**: 2nd conversion of the sequence  
These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.  
When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.  
Refer to SQ8[3:0] for a definition of channel selection.  
*Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).*

Bits 3:0 **SQ1[3:0]**: 1st conversion of the sequence  
These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.  
When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.  
Refer to SQ8[3:0] for a definition of channel selection.  
*Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).*

14.12.11 **ADC watchdog threshold register (ADC_AWD3TR)**

Address offset: 0x2C

Reset value: 0x0FFF 0000
### ADC data register (ADC_DR)

**Address offset:** 0x40

**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>DATA[15:0]</td>
<td>r</td>
<td>Converted data. These bits are read-only. They contain the conversion result from the last converted channel. The data are left- or right-aligned as shown in Figure 42: Data alignment and resolution (oversampling disabled: OVSE = 0) on page 323. Just after a calibration is complete, DATA[6:0] contains the calibration factor.</td>
</tr>
<tr>
<td>15:12</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:0</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Bits 27:16** HT3[11:0]: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog. Refer to Section 14.7: Analog window watchdog (AWD1EN, AWD1SGL, AWD1CH, ADC_AWDxCR, ADC_AWDxTR) on page 328.

**Bits 15:12** Reserved, must be kept at reset value.

**Bits 11:0** LT3[11:0]: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog. Refer to Section 14.7: Analog window watchdog (AWD1EN, AWD1SGL, AWD1CH, ADC_AWDxCR, ADC_AWDxTR) on page 328.
### 14.12.13 ADC Analog Watchdog 2 Configuration register (ADC_AWD2CR)

Address offset: 0xA0
Reset value: 0x0000 0000

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **AWD2CH[18:0]:** Analog watchdog channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 2 (AWD2).

0: ADC analog channel-x is not monitored by AWD2
1: ADC analog channel-x is monitored by AWD2

Note: The channels selected through ADC_AWD2CR must be also configured into the ADC_CHSELR registers. Refer to SQ8[3:0] for a definition of channel selection. The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

### 14.12.14 ADC Analog Watchdog 3 Configuration register (ADC_AWD3CR)

Address offset: 0xA4
Reset value: 0x0000 0000

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **AWD3CH[18:0]:** Analog watchdog channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 3 (AWD3).

0: ADC analog channel-x is not monitored by AWD3
1: ADC analog channel-x is monitored by AWD3

Note: The channels selected through ADC_AWD3CR must be also configured into the ADC_CHSELR registers. Refer to SQ8[3:0] for a definition of channel selection. The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).
14.12.15 ADC Calibration factor (ADC_CALFACT)

Address offset: 0xB4
Reset value: 0x0000 0000

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

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 CALFACT[6:0]: Calibration factor
These bits are written by hardware or by software.
- Once a calibration is complete, they are updated by hardware with the calibration factors.
- Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it will then be applied once a new calibration is launched.
- Just after a calibration is complete, DATA[6:0] contains the calibration factor.

Note: Software is allowed to write these bits only when ADEN=1 (ADC is enabled and no calibration is ongoing and no conversion is ongoing). Refer to SQ8[3:0] for a definition of channel selection.

14.12.16 ADC common configuration register (ADC_CCR)

Address offset: 0x308
Reset value: 0x0000 0000

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

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

rw rw rw rw rw rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: $V_{\text{BAT}}$ enable
This bit is set and cleared by software to enable/disable the $V_{\text{BAT}}$ channel.
- 0: $V_{\text{BAT}}$ channel disabled, DAC\_OUT2 connected to ADC channel 14
- 1: $V_{\text{BAT}}$ channel enabled

*Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing)*

Bit 23 **TSEN**: Temperature sensor enable
This bit is set and cleared by software to enable/disable the temperature sensor.
- 0: Temperature sensor disabled, DAC\_OUT1 connected to ADC channel 12
- 1: Temperature sensor enabled

*Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing)*

Bit 22 **VREFEN**: $V_{\text{REFINT}}$ enable
This bit is set and cleared by software to enable/disable the $V_{\text{REFINT}}$.
- 0: $V_{\text{REFINT}}$ disabled
- 1: $V_{\text{REFINT}}$ enabled

*Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing)*

Bits 21:18 **PRESC[3:0]**: ADC prescaler
Set and cleared by software to select the frequency of the clock to the ADC.
- 0000: input ADC clock not divided
- 0001: input ADC clock divided by 2
- 0010: input ADC clock divided by 4
- 0011: input ADC clock divided by 6
- 0100: input ADC clock divided by 8
- 0101: input ADC clock divided by 10
- 0110: input ADC clock divided by 12
- 0111: input ADC clock divided by 16
- 1000: input ADC clock divided by 32
- 1001: input ADC clock divided by 64
- 1010: input ADC clock divided by 128
- 1011: input ADC clock divided by 256
- Other: Reserved

*Note: Software is allowed to write these bits only when the ADC is disabled (ADCAL=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).*

Bits 17:0 Reserved, must be kept at reset value.
### 14.12.17 ADC register map

The following table summarizes the ADC registers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>ADC_ISR</td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000 000 000 000</td>
</tr>
<tr>
<td>0x04</td>
<td>ADC_IER</td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000 000 000 000</td>
</tr>
<tr>
<td>0x08</td>
<td>ADC_CR</td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000 000 000 000</td>
</tr>
<tr>
<td>0x0C</td>
<td>ADC_CFRG1</td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000 000 000 000</td>
</tr>
<tr>
<td>0x10</td>
<td>ADC_CFRG2</td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000 000 000 000</td>
</tr>
<tr>
<td>0x14</td>
<td>ADC_SMPR</td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000 000 000 000</td>
</tr>
<tr>
<td>0x18</td>
<td>ADC_AWD1TR</td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>111 111 111 111</td>
</tr>
<tr>
<td>0x20</td>
<td>ADC_AWD2TR</td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000 000 000 000</td>
</tr>
<tr>
<td>0x24</td>
<td>ADC_CHSEL128</td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000 000 000 000</td>
</tr>
<tr>
<td>0x28</td>
<td>ADC_CHSEL128</td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000 000 000 000</td>
</tr>
<tr>
<td>0x2C</td>
<td>ADC_AWD3TR</td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>111 111 111 111</td>
</tr>
</tbody>
</table>

Table 68. ADC register map and reset values
Refer to Section 2.2 on page 54 for the register boundary addresses.
15 Digital-to-analog converter (DAC)

15.1 Introduction
The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. The DAC features two output channels, each with its own converter. In dual DAC channel mode, conversions could be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, VREF+ (shared with others analog peripherals) is available for better resolution. An internal reference can also be set on the same input. Refer to voltage reference buffer (VREFBUF) section.

The DAC_OUTx pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on chip peripheral. The DAC output buffer can be optionally enabled to allow a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the Sample and Hold mode.

15.2 DAC main features
The DAC main features are the following (see Figure 57: Dual-channel DAC block diagram)
- One DAC interface, maximum two output channels
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Each DAC output can be disconnected from the DAC_OUTx output pin
- DAC output connection to on chip peripherals
- Sample and Hold mode for low power operation in Stop mode
- Input voltage reference, VREF+

Figure 57 shows the block diagram of a DAC channel and Table 70 gives the pin description.
15.3 DAC implementation

Table 69. DAC implementation

<table>
<thead>
<tr>
<th>DAC features</th>
<th>DAC1&lt;sup&gt;(1)&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual channel</td>
<td>X</td>
</tr>
<tr>
<td>I/O connection</td>
<td>DAC1_OUT1 on PA4, DAC1_OUT2 on PA5</td>
</tr>
</tbody>
</table>

1. There is no DAC on STM32G031xx and STM32G041xx.
15.4  DAC functional description

15.4.1  DAC block diagram

Figure 57. Dual-channel DAC block diagram

1. MODEx bits in the DAC_MCR control the output mode and allow switching between the Normal mode in buffer/unbuffered configuration and the Sample and Hold mode.

2. Refer to Section 15.3: DAC implementation for channel2 availability.
15.4.2 DAC pins and internal signals

The DAC includes:

- Up to two output channels
- The DAC_OUTx can be disconnected from the output pin and used as an ordinary GPIO
- The dac_outx can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered
- Sample and Hold block and registers operational in Stop mode, using LSI clock source for static conversion

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DAC_OUTx output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The Sample and Hold block and its associated registers can run in Stop mode using the LSI clock source.

Table 70. DAC input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREF+</td>
<td>Input, analog reference positive</td>
<td>The higher/positive reference voltage for the DAC, $V_{REF+} \leq V_{DDA_{max}}$ (refer to datasheet)</td>
</tr>
<tr>
<td>VDD</td>
<td>Input, analog supply</td>
<td>Analog power supply</td>
</tr>
<tr>
<td>VSS</td>
<td>Input, analog supply ground</td>
<td>Ground for analog power supply</td>
</tr>
<tr>
<td>DAC_OUTx</td>
<td>Analog output signal</td>
<td>DAC channelx analog output</td>
</tr>
</tbody>
</table>

Table 71. DAC internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dac_ch1_dma</td>
<td>Bidirectional</td>
<td>DAC channel1 DMA request</td>
</tr>
<tr>
<td>dac_ch2_dma</td>
<td>Bidirectional</td>
<td>DAC channel2 DMA request</td>
</tr>
<tr>
<td>dac_ch1_trg[0:15]</td>
<td>Inputs</td>
<td>DAC channel1 trigger inputs</td>
</tr>
<tr>
<td>dac_ch2_trg[0:15]</td>
<td>Inputs</td>
<td>DAC channel2 trigger inputs</td>
</tr>
<tr>
<td>dac_unr_it</td>
<td>Output</td>
<td>DAC underrun interrupt</td>
</tr>
<tr>
<td>dac_pclk</td>
<td>Input</td>
<td>DAC peripheral clock</td>
</tr>
<tr>
<td>dac_out1</td>
<td>Analog output</td>
<td>DAC channel1 output for on-chip peripherals</td>
</tr>
<tr>
<td>dac_out2</td>
<td>Analog output</td>
<td>DAC channel2 output for on-chip peripherals</td>
</tr>
</tbody>
</table>
15.4.3 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a tWAKEUP startup time.

*Note:* The ENx bit enables the analog DAC channel x only. The DAC channel x digital interface is enabled even if the ENx bit is reset.

15.4.4 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel
  
  There are three possibilities:
  
  - 8-bit right alignment: the software has to load data into the DAC_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)
  
  - 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)
  
  - 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

  Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DHRx (data holding registers, which are internal non-memory-mapped registers). The DHRx register is then loaded into the DORx register either automatically, by software trigger or by an external event trigger.

- Dual DAC channels (when available)
  
  There are three possibilities:
  
  - 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)
  
  - 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)
  
  - 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

  Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC_DOR1.
and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

**Figure 59. Data registers in dual DAC channel mode**

<table>
<thead>
<tr>
<th>31</th>
<th>24</th>
<th>15</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>8-bit right aligned</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12-bit left aligned</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12-bit right aligned</td>
<td></td>
</tr>
</tbody>
</table>

**15.4.5 DAC conversion**

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write operation to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one dac_pclk clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_pclk clock cycles after the trigger signal.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time \( t_{SETTLING} \) that depends on the power supply voltage and the analog output load.

**Figure 60. Timing diagram for conversion with trigger disabled TEN = 0**

**15.4.6 DAC output voltage**

Digital inputs are converted to output voltages on a linear conversion between 0 and \( V_{REF+} \).

The analog output voltages on each DAC channel pin are determined by the following equation:

\[
DACoutput = V_{REF} \times \frac{DOR}{4096}
\]
15.4.7 DAC trigger selection

If the TENx control bit is set, conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[3:0] control bits determine which out of 16 possible events will trigger conversion as shown in bits TSEL1[3:0] and TSEL2[3:0] in Table 72: DAC trigger selection.

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three dac_pclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: TSELx[3:0] bit cannot be changed when the ENx bit is set.

When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one dac_pclk clock cycle.

Table 72. DAC trigger selection(1)

<table>
<thead>
<tr>
<th>Source</th>
<th>Type</th>
<th>TSELx[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWTRIG</td>
<td>Software control bit</td>
<td>0000</td>
</tr>
<tr>
<td>TIM1_TRGO</td>
<td>Internal signal from on-chip timers</td>
<td>0001</td>
</tr>
<tr>
<td>TIM2_TRGO</td>
<td>Internal signal from on-chip timers</td>
<td>0010</td>
</tr>
<tr>
<td>TIM3_TRGO</td>
<td>Internal signal from on-chip timers</td>
<td>0011</td>
</tr>
<tr>
<td>Reserved</td>
<td>Internal signal from on-chip timers</td>
<td>0100</td>
</tr>
<tr>
<td>TIM6_TRGO</td>
<td>Internal signal from on-chip timers</td>
<td>0101</td>
</tr>
<tr>
<td>TIM7_TRGO</td>
<td>Internal signal from on-chip timers</td>
<td>0110</td>
</tr>
<tr>
<td>Reserved</td>
<td>Internal signal from on-chip timers</td>
<td>0111</td>
</tr>
<tr>
<td>TIM15_TRGO</td>
<td>Internal signal from on-chip timers</td>
<td>1000</td>
</tr>
<tr>
<td>TIM3_TRGO</td>
<td>-</td>
<td>1001</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>1010</td>
</tr>
<tr>
<td>LPTIM1_OUT</td>
<td>Internal signal from on-chip timers</td>
<td>1011</td>
</tr>
<tr>
<td>LPTIM2_OUT</td>
<td>Internal signal from on-chip timers</td>
<td>1100</td>
</tr>
<tr>
<td>EXTI9</td>
<td>External pin</td>
<td>1101</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>1110</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>1111</td>
</tr>
</tbody>
</table>

1. TIM3_TRGO can be selected either by setting TSELx bits to 0b0011 or to 0b1001.
15.4.8 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC_DHRx register is transferred into the DAC_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit should be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC_DHRx to DAC_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHRx before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

The software should clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software should modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion could be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

15.4.9 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01”. The preloaded value in LFSR is 0xAAA. This register is updated three dac_pclk clock cycles after each trigger event, following a specific calculation algorithm.
The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

If LFSR is 0x0000, a '1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.
15.4.10 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_pclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is transferred into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

**Figure 63. DAC triangle wave generation**

**Figure 64. DAC conversion (SW trigger enabled) with triangle wave generation**

*Note: The DAC trigger must be enabled for triangle wave generation by setting the TENx bit in the DAC_CR register.

*The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.*
15.4.11 DAC channel modes

Each DAC channel can be configured in Normal mode or Sample and Hold mode. The output buffer can be enabled to allow a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In Normal mode, there are four combinations, by changing the buffer state and by changing the DAC_OUTx pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register should be:
- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODEx[2:0] bits in DAC_MCR register should be:
- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

Sample and Hold mode

In sample and Hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A new stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the low-speed clock (LSI) in addition to the dac_pclk clock, allowing to use the DAC channels in deep low power modes such as Stop mode.

The sample/hold mode operations can be divided into 3 phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEx[9:0] bits in DAC_SHSRx register. During the write of the TSAMPLEx[9:0] bits; the BWSTx bit in DAC_SR register is set to 1 to synchronize between both clocks domains (APB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation.

2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC_SHHR register.

3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC_SHRR register.

The timings for the three phases above are in units of LSI clocks. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI ~32 KHz is selected:

- 12 cycles are required for sample phase: TSAMPLEx[9:0] = 11,
- 62 cycles are required for hold phase: THOLDx[9:0] = 62,
- and 4 cycles are required for refresh period: TREFRESHx[7:0] = 4.
In this example, the power consumption is reduced by almost a factor of 15 versus Normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

### Table 73. Sample and refresh timings

<table>
<thead>
<tr>
<th>Buffer State</th>
<th>$t_{SAMP}^{(1) (2)}$</th>
<th>$t_{REFRESH}^{(2) (3)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>$7 \mu s + (10*R_{BON}*C_{SH})$</td>
<td>$7 \mu s + (R_{BON}<em>C_{SH})</em>\ln(2*N_{LSB})$</td>
</tr>
<tr>
<td>Disable</td>
<td>$3 \mu s + (10*R_{BOFF}*C_{SH})$</td>
<td>$3 \mu s + (R_{BOFF}<em>C_{SH})</em>\ln(2*N_{LSB})$</td>
</tr>
</tbody>
</table>

1. In the above formula the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8 bits resolution, the settling time is 7 constant time.
2. $C_{SH}$ is the capacitor in Sample and Hold mode.
3. The tolerated voltage drop during the hold phase "$V_d$" is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires $\ln(2*N_{LSB})$ constant time of the DAC.

### Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only. Please refer to the product datasheet for product data.

$C_{SH} = 100 \, nF$
$V_{DD} = 3.0 \, V$

**Sampling phase:**

\[ t_{SAMP} = 7 \mu s + (10*2000*100*10^{-9}) = 2.007 \, ms \]

(where $R_{BON} = 2 \, k\Omega$)

**Refresh phase:**

\[ t_{REFRESH} = 7 \mu s + (2000*100*10^{-9}) \times \ln(2*10) = 606.1 \, \mu s \]

(where $N_{LSB} = 10$ (10 LSB drop during the hold phase))

**Hold phase:**

\[ D_v = I_{leak} \times t_{hold} / C_{SH} = 0.0073 \, V \] (10 LSB of 12bit at 3 V)

$I_{leak} = 150 \, nA$ (worst case on the IO leakage on all the temperature range)

\[ t_{hold} = 0.0073 \times 100 \times 10^{-9} / (150 \times 10^{-9}) = 4.867 \, ms \]
Like in Normal mode, the Sample and Hold mode has different configurations.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register should be:

- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, The MODEx[2:0] bits in DAC_MCR register should be:

- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals

When MODEx[2:0] bits in DAC_MCR register is equal to 111. An internal capacitor, C_{Lint}, will hold the voltage output of the DAC Core and then drive it to on-chip peripherals.

All Sample and Hold phases are interruptible and any change in DAC_DHRx will trigger immediately a new sample phase.

Table 74. Channel output modes summary

<table>
<thead>
<tr>
<th>MODEx[2:0]</th>
<th>Mode</th>
<th>Buffer</th>
<th>Output connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>Normal mode</td>
<td>Enabled</td>
<td>Connected to external pin</td>
</tr>
<tr>
<td>0 0 1</td>
<td></td>
<td></td>
<td>Connected to external pin and to on chip-peripherals (ex, comparators)</td>
</tr>
<tr>
<td>0 1 0</td>
<td></td>
<td>Disabled</td>
<td>Connected to external pin</td>
</tr>
<tr>
<td>0 1 1</td>
<td></td>
<td></td>
<td>Connected to on chip peripherals (ex, comparators)</td>
</tr>
</tbody>
</table>
The transfer function for an N-bit digital-to-analog converter (DAC) is:

\[ V_{\text{out}} = \left( \frac{D}{2^{N-1}} \right) \times G \times V_{\text{ref}} + V_{\text{OS}} \]

Where \( V_{\text{OUT}} \) is the analog output, \( D \) is the digital input, \( G \) is the gain, \( V_{\text{ref}} \) is the nominal full-scale voltage, and \( V_{\text{OS}} \) is the offset voltage. For an ideal DAC channel, \( G = 1 \) and \( V_{\text{OS}} = 0 \).

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the \( V_{\text{OS}} \), a calibration is required by a trimming technique.

The calibration is only valid when the DAC channel is operating with buffer enabled (MODEx[2:0] = 000b or 001b or 100b or 101b). If applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output will be disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer will act as a comparator, to sense the middle-code value 0x800 and compare it to VREF+/2 signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAGx bit).

Two calibration techniques are provided:

- Factory trimming (always enabled)
  The DAC buffer offset is factory trimmed. The default value of OTRIMx[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.

- User trimming
  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when \( V_{\text{DD}} \) voltage, temperature, VREF+ values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the Nominal factory trimming conditions

In addition, when \( V_{\text{DD}} \) is removed (example the device enters in STANDBY or VBAT modes) the calibration is required.

The steps to perform a user trimming calibration are as below:
1. If the DAC channel is active, write 0 to ENx bit in DAC_CR to disable the channel.

2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, \( \text{MODEx}[2:0] = 000b \) or \( 001b \) or \( 100b \) or \( 101b \).

3. Start the DAC channel calibration, by setting the CENx bit in DAC_CR register to 1.

4. Apply a trimming algorithm:
   a) Write a code into OTRIMx[4:0] bits, starting by 00000b.
   b) Wait for \( t_{\text{TRIM}} \) delay.
   c) Check if CAL_FLAGx bit in DAC_SR is set to 1.
   d) If CAL_FLAGx is set to 1 the trimming code OTRIMx[4:0] is found and will be used during operation to compensate the output value, else increment OTRIMx[4:0] and repeat sub-steps from (a) to (d) again.

   The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way.

   The commutation/toggle of CAL_FLAGx bit indicates that the offset is correctly compensated and the corresponding trim code must be kept in the OTRIMx[4:0] bits in DAC_CCR register.

   \textbf{Note:} A \( t_{\text{TRIM}} \) delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL_FLAGx bit in DAC_SR register in order to get a correct value. This parameter is specified into datasheet electrical characteristics section.

   \textit{If the VDD, VREF+ and temperature conditions will not change during the device operation while it enters more often in standby and VBAT mode, the software may store the OTRIMx[4:0] bits found in the first user calibration in the flash or in back-up registers. Then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.}

   \textit{When CENx bit is set, it is not allowed to set ENx bit.}

15.4.13 \textbf{Dual DAC channel conversion modes (if available)}

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

11 conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

All modes are described in the paragraphs below.

**Independent trigger without wave generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.

2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.

3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).
When a DAC channel 1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_pclk clock cycles later).

When a DAC channel 2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three dac_pclk clock cycles later).

**Independent trigger with single LFSR generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel 1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel 2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). Then the LFSR2 counter is updated.

**Independent trigger with different LFSR generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel 1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel 2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). Then the LFSR2 counter is updated.

**Independent trigger with single triangle generation**

To configure the DAC in this conversion mode, the following sequence is required:
1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel 1 trigger arrives, the DAC channel 1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel 1 triangle counter is then updated.

When a DAC channel 2 trigger arrives, the DAC channel 2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The DAC channel 2 triangle counter is then updated.

**Independent trigger with different triangle generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1[ and TSEL2 bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel 1 trigger arrives, the DAC channel 1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel 1 triangle counter is then updated.

When a DAC channel 2 trigger arrives, the DAC channel 2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The DAC channel 2 triangle counter is then updated.

**Simultaneous software start**

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

In this configuration, one dac_pclk clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

**Simultaneous trigger without wave generation**

To configure the DAC in this conversion mode, the following sequence is required:
1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three dac_pclk clock cycles).

**Simultaneous trigger with single LFSR generation**

1. To configure the DAC in this conversion mode, the following sequence is required:
2. Set the two DAC channel trigger enable bits TEN1 and TEN2.
3. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
4. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
5. Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The LFSR2 counter is then updated.

**Simultaneous trigger with different LFSR generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The LFSR2 counter is then updated.

**Simultaneous trigger with single triangle generation**

To configure the DAC in this conversion mode, the following sequence is required:
1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The DAC channel2 triangle counter is then updated.

**Simultaneous trigger with different triangle generation**

To configure the DAC in this conversion mode, the following sequence is required:
1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). Then the DAC channel2 triangle counter is updated.
15.5 DAC low-power modes

Table 75. Effect of low-power modes on DAC

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect, DAC used with DMA</td>
</tr>
<tr>
<td>Stop 0</td>
<td>DAC remains active with a static value, if Sample and Hold mode is selected using LSI clock</td>
</tr>
<tr>
<td>Standby</td>
<td>The DAC peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.</td>
</tr>
<tr>
<td>Shutdown</td>
<td></td>
</tr>
</tbody>
</table>

15.6 DAC interrupts

Table 76. DAC interrupts

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA underrun</td>
<td>DMAUDRx</td>
<td>DMAUDRIEx</td>
</tr>
</tbody>
</table>
15.7  DAC registers

Refer to Section 1 on page 50 for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

15.7.1  DAC control register (DAC_CR)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>CEN2</td>
<td>DMAU</td>
<td>DMAE</td>
<td>MAMP2</td>
<td>WAVE2</td>
<td>TSEL2</td>
<td>TSEL1</td>
<td>TSEL0</td>
<td>TEN2</td>
<td>EN2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DRIE2</td>
<td>N2</td>
<td>[3:0]</td>
<td>[1:0]</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>

Bit 31  Reserved, must be kept at reset value.

Bit 30  **CEN2**: DAC channel2 calibration enable

This bit is set and cleared by software to enable/disable DAC channel2 calibration, it can be written only if EN2 bit is set to 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel2 in Normal operating mode
1: DAC channel2 in calibration mode

Note:  *This bit is available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.*

Bit 29  **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled
1: DAC channel2 DMA underrun interrupt enabled

Note:  *This bit is available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.*

Bit 28  **DMAEN2**: DAC channel2 DMA enable

This bit is set and cleared by software.

0: DAC channel2 DMA mode disabled
1: DAC channel2 DMA mode enabled

Note:  *This bit is available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.*
Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

- 0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
- 0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
- 0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
- 0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
- 0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
- 0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
- 0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
- 0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
- 1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
- 1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
- 1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
- 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

*Note:* These bits are available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable

These bits are set/reset by software.

- 00: wave generation disabled
- 01: Noise wave generation enabled
- 1x: Triangle wave generation enabled

*Note:* Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)

*These bits are available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.*

Bits 21:18 **TSEL2[3:0]**: DAC channel2 trigger selection

These bits select the external event used to trigger DAC channel2

Refer to the trigger selection tables for the details on trigger configuration and mapping.

*Note:* Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).

*These bits are available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.*

Bit 17 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

- 0: DAC channel2 trigger disabled and data written into the DAC_DHR2 register are transferred one dac_pclk clock cycle later to the DAC_DOR2 register
- 1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three dac_pclk clock cycles later to the DAC_DOR2 register

*Note:* When software trigger is selected, the transfer from the DAC_DHR2 register to the DAC_DOR2 register takes only one dac_pclk clock cycle.

*These bits are available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.*

Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.

- 0: DAC channel2 disabled
- 1: DAC channel2 enabled

*Note:* These bits are available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.

Bit 15 Reserved, must be kept at reset value.
Bit 14 **CEN1**: DAC channel1 calibration enable
This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1=0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.
0: DAC channel1 in Normal operating mode
1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable
This bit is set and cleared by software.
0: DAC channel1 DMA Underrun Interrupt disabled
1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable
This bit is set and cleared by software.
0: DAC channel1 DMA mode disabled
1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector
These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.
0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable
These bits are set and cleared by software.
00: wave generation disabled
01: Noise wave generation enabled
1x: Triangle wave generation enabled
Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).
Bits 5:2 TSEL[3:0]: DAC channel1 trigger selection  
These bits select the external event used to trigger DAC channel1  
Refer to the trigger selection tables for the details on trigger configuration and mapping.  
*Note:* Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 TEN1: DAC channel1 trigger enable  
This bit is set and cleared by software to enable/disable DAC channel1 trigger.  
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_pclk clock cycle later to the DAC_DOR1 register  
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_pclk clock cycles later to the DAC_DOR1 register  
*Note:* When software trigger is selected, the transfer from the DAC_DHR1 register to the DAC_DOR1 register takes only one dac_pclk clock cycle.

Bit 0 EN1: DAC channel1 enable  
This bit is set and cleared by software to enable/disable DAC channel1.  
0: DAC channel1 disabled  
1: DAC channel1 enabled

### 15.7.2 DAC software trigger register (DAC_SWTRGR)

Address offset: 0x04  
Reset value: 0x0000 0000

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

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

Bit 1 SWTRIG2: DAC channel2 software trigger  
This bit is set by software to trigger the DAC in software trigger mode.  
0: No trigger  
1: Trigger  
*Note:* This bit is cleared by hardware (one dac_pclk clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.  
This bit is available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.

Bit 0 SWTRIG1: DAC channel1 software trigger  
This bit is set by software to trigger the DAC in software trigger mode.  
0: No trigger  
1: Trigger  
*Note:* This bit is cleared by hardware (one dac_pclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.
15.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08
Reset value: 0x0000 0000

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 DACC1DHR[11:0]: DAC channel1 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel1.

15.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C
Reset value: 0x0000 0000

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 DACC1DHR[11:0]: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.
15.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10
Reset value: 0x0000 0000

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

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

Bits 7:0 DACC1DHR[7:0]: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.

15.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

This register is available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.

Address offset: 0x14
Reset value: 0x0000 0000

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 DACC2DHR[11:0]: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2.
15.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

This register is available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.

Address offset: 0x18
Reset value: 0x0000 0000

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 DAC2DHR[11:0]: DAC channel2 12-bit left-aligned data
These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

15.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

This register is available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.

Address offset: 0x1C
Reset value: 0x0000 0000

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

Bits 7:0 DAC2DHR[7:0]: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.
15.7.9  **Dual DAC 12-bit right-aligned data holding register**  
(DAC_DHR12RD)

Address offset: 0x20  
Reset value: 0x0000 0000

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

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

Bits 27:16  **DAC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 15:12  Reserved, must be kept at reset value.

Bits 11:0  **DAC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

15.7.10  **Dual DAC 12-bit left aligned data holding register**  
(DAC_DHR12LD)

Address offset: 0x24  
Reset value: 0x0000 0000

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

Bits 31:20  **DAC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 19:16  Reserved, must be kept at reset value.

Bits 15:4  **DAC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0  Reserved, must be kept at reset value.
15.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28
Reset value: 0x0000 0000

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

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

<table>
<thead>
<tr>
<th>DACC2DHR[7:0]</th>
<th>DACC1DHR[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

15.7.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C
Reset value: 0x0000 0000

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

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

<table>
<thead>
<tr>
<th>DACC1DOR[11:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r</td>
</tr>
</tbody>
</table>

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.
15.7.13 DAC channel2 data output register (DAC_DOR2)

This register is available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.

Address offset: 0x30
Reset value: 0x0000 0000

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

DACC2DOR[11:0]

- Bits 31:12: Reserved, must be kept at reset value.
- Bits 11:0: DACC2DOR[11:0]: DAC channel2 data output
  These bits are read-only, they contain data output for DAC channel2.

15.7.14 DAC status register (DAC_SR)

Address offset: 0x34
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td></td>
<td>rc_w1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td></td>
<td>rc_w1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bit 31 **BWST2**: DAC channel2 busy writing sample time flag
This bit is systematically set just after Sample and Hold mode enable. It is set each time the software writes the register DAC_SHSR2. It is cleared by hardware when the write operation of DAC_SHSR2 is complete. (It takes about 3 LSI periods of synchronization).

0: There is no write operation of DAC_SHSR2 ongoing: DAC_SHSR2 can be written
1: There is a write operation of DAC_SHSR2 ongoing: DAC_SHSR2 cannot be written

*Note: This bit is available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.*

Bit 30 **CAL_FLAG2**: DAC channel2 calibration offset status
This bit is set and cleared by hardware

0: Calibration trimming value is lower than the offset correction value
1: Calibration trimming value is equal or greater than the offset correction value

*Note: This bit is available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.*

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag
This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel2
1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).

*Note: This bit is available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.*

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC channel1 busy writing sample time flag
This bit is systematically set just after Sample and Hold mode enable and is set each time the software writes the register DAC_SHSR1. It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI periods of synchronization).

0: There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written
1: There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC channel1 calibration offset status
This bit is set and cleared by hardware

0: Calibration trimming value is lower than the offset correction value
1: Calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag
This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1
1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bits 10:0 Reserved, must be kept at reset value.
15.7.15  **DAC calibration control register (DAC_CCR)**

Address offset: 0x38
Reset value: 0x00XX 00XX

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

Bits 31:21  Reserved, must be kept at reset value.

Bits 20:16  **OTRIM2[4:0]**: DAC channel2 offset trimming value

These bits are available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.

Bits 15:5  Reserved, must be kept at reset value.

Bits 4:0  **OTRIM1[4:0]**: DAC channel1 offset trimming value

15.7.16  **DAC mode control register (DAC_MCR)**

Address offset: 0x3C
Reset value: 0x0000 0000

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

Bits 31:26  Reserved, must be kept at reset value.

Bit 25  Reserved, must be kept at reset value.

Bit 24  Reserved, must be kept at reset value.

Bits 23:19  Reserved, must be kept at reset value.
Bits 18:16 **MODE2[2:0]**: DAC channel2 mode
These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2=0 and bit CEN2 =0 in the DAC_CR register). If EN2=1 or CEN2 =1 the write operation is ignored.
They can be set and cleared by software to select the DAC channel2 mode:
– DAC channel2 in Normal mode
  000: DAC channel2 is connected to external pin with Buffer enabled
  001: DAC channel2 is connected to external pin and to on chip peripherals with buffer enabled
  010: DAC channel2 is connected to external pin with buffer disabled
  011: DAC channel2 is connected to on chip peripherals with Buffer disabled
– DAC channel2 in Sample and Hold mode
  100: DAC channel2 is connected to external pin with Buffer enabled
  101: DAC channel2 is connected to external pin and to on chip peripherals with Buffer enabled
  110: DAC channel2 is connected to external pin and to on chip peripherals with Buffer disabled
  111: DAC channel2 is connected to on chip peripherals with Buffer disabled

*Note:* This register can be modified only when EN2=0.
Refer to Section 15.3: DAC implementation for the availability of DAC channel2.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

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

Bits 2:0 **MODE1[2:0]**: DAC channel1 mode
These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1=0 and bit CEN1 =0 in the DAC_CR register). If EN1=1 or CEN1 =1 the write operation is ignored.
They can be set and cleared by software to select the DAC channel1 mode:
– DAC channel1 in Normal mode
  000: DAC channel1 is connected to external pin with Buffer enabled
  001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
  010: DAC channel1 is connected to external pin with Buffer disabled
  011: DAC channel1 is connected to on chip peripherals with Buffer disabled
– DAC channel1 in sample & hold mode
  100: DAC channel1 is connected to external pin with Buffer enabled
  101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
  110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled
  111: DAC channel1 is connected to on chip peripherals with Buffer disabled

*Note:* This register can be modified only when EN1=0.
15.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1)

Address offset: 0x40
Reset value: 0x0000 0000

| Bits 31:10 | Reserved, must be kept at reset value. |
| Bits 9:0   | **TSAMPLE1[9:0]**: DAC channel1 sample time (only valid in Sample and Hold mode) |
|           | These bits can be written when the DAC channel1 is disabled or also during normal operation. In the latter case, the write can be done only when BWST1 of DAC_SR register is low. If BWST1=1, the write operation is ignored. |

**Note:** It represents the number of LSI clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x LSI clock period.

15.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2)

This register is available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.

Address offset: 0x44
Reset value: 0x0000 0000

| Bits 31:10 | Reserved, must be kept at reset value. |
| Bits 9:0   | **TSAMPLE2[9:0]**: DAC channel2 sample time (only valid in Sample and Hold mode) |
|           | These bits can be written when the DAC channel2 is disabled or also during normal operation. In the latter case, the write can be done only when BWST2 of DAC_SR register is low. If BWST2=1, the write operation is ignored. |

**Note:** It represents the number of LSI clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x LSI clock period.
15.7.19 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48
Reset value: 0x0001 0001

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 \textbf{THOLD2}[9:0]: DAC channel2 hold time (only valid in Sample and Hold mode).
Hold time = (THOLD[9:0]) x LSI clock period
Note: This register can be modified only when \text{EN2}=0.
These bits are available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 \textbf{THOLD1}[9:0]: DAC channel1 hold time (only valid in Sample and Hold mode)
Hold time = (THOLD[9:0]) x LSI clock period
Note: This register can be modified only when \text{EN1}=0.

Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit \text{ENx}=0 and bit \text{CENx}=0 in the DAC_CR register). If \text{ENx}=1 or \text{CENx}=1 the write operation is ignored.

15.7.20 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C
Reset value: 0x0001 0001

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

Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit \text{ENx}=0 and bit \text{CENx}=0 in the DAC_CR register). If \text{ENx}=1 or \text{CENx}=1 the write operation is ignored.
Bits 31:24  Reserved, must be kept at reset value.

Bits 23:16  **TREFRESH2[7:0]**: DAC channel2 refresh time (only valid in Sample and Hold mode)
            Refresh time= (TREFRESH[7:0]) x LSI clock period
            
            Note:  This register can be modified only when EN2=0.
            These bits are available only on dual-channel DACs. Refer to Section 15.3: DAC implementation.

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:0  **TREFRESH1[7:0]**: DAC channel1 refresh time (only valid in Sample and Hold mode)
          Refresh time= (TREFRESH[7:0]) x LSI clock period
          
          Note:  This register can be modified only when EN1=0.

Note:  These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit ENx=0 and bit CENx=0 in the DAC_CR register). If ENx=1 or CENx=1 the write operation is ignored.
15.7.21  DAC register map

*Table 77* summarizes the DAC registers.

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | DAC_CR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | DAC_SWTRGR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x08   | DAC_DHR12R1   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0C   | DAC_DHR12L1   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x10   | DAC_DHR8R1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x14   | DAC_DHR8R2    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1C   | DAC_DHR8R2    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x20   | DAC_DHR12RD   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x24   | DAC_DHR12LD   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x28   | DAC_DHR8RD    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x2C   | DAC_DOR1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x30   | DAC_DOR2      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x34   | DAC_SR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Table 77. DAC register map and reset values

- **Offset**: Address of the register in hexadecimal form.
- **Register name**: Name of the register.
- **Reset value**: Initial value of the register upon reset.

For example, the reset value for `DAC_CR` is 00000000000000000000000000000000.
Refer to Section 2.2 on page 54 for the register boundary addresses.
16 Voltage reference buffer (VREFBUF)

16.1 Introduction
The STM32G0x1 devices embed a voltage reference buffer which can be used as voltage reference for ADC, DAC and also as voltage reference for external components through the VREF+ pin. When the VREF+ pin is double-bonded with VDDA pin in a package, the voltage reference buffer is not available and must be kept disabled (refer to datasheet for packages pinout description).

16.2 VREFBUF functional description
The internal voltage reference buffer supports two voltages\(^{(a)}\), which are configured with VRS bits in the VREFBUF_CSR register:
- VRS = 0: V_{REF\_OUT1} around 2.048 V.
- VRS = 1: V_{REF\_OUT2} around 2.5 V.

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

<table>
<thead>
<tr>
<th>ENVR</th>
<th>HIZ</th>
<th>VREF buffer configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>VREFBUF buffer OFF:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREF+ pin pulled-down to VSSA</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>External voltage reference mode (default value):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREFBUF buffer OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREF+ pin input mode</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Internal voltage reference mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREFBUF buffer ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREF+ pin connected to VREFBUF buffer output</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Hold mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREFBUF buffer OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREF+ pin floating. The voltage is held with the external capacitor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VRR detection disabled and VRR bit keeps last state</td>
</tr>
</tbody>
</table>

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

\(^{(a)}\) The minimum \(V_{DDA}\) voltage depends on VRS setting, refer to the product datasheet.
16.3 VREFBUFF registers

16.3.1 VREFBUFF control and status register (VREFBUFF_CSR)

Address offset: 0x00
Reset value: 0x0000 0002

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>31:4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Res.</td>
<td>VRR</td>
<td>VRS</td>
<td>HIZ</td>
<td>ENVR</td>
</tr>
</tbody>
</table>

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

Bit 3 **VRR**: Voltage reference buffer ready
0: the voltage reference buffer output is not ready.
1: the voltage reference buffer output reached the requested level.

Bit 2 **VRS**: Voltage reference scale
This bit selects the value generated by the voltage reference buffer.
0: Voltage reference set to $V_{REF\_OUT1}$ (around 2.048 V).
1: Voltage reference set to $V_{REF\_OUT2}$ (around 2.5 V).

Bit 1 **HIZ**: High impedance mode
This bit controls the analog switch to connect or not the $V_{REF+}$ pin.
0: $V_{REF+}$ pin is internally connected to the voltage reference buffer output.
1: $V_{REF+}$ pin is high impedance.
Refer to Table 78: VREF buffer modes for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR**: Voltage reference buffer mode enable
This bit is used to enable the voltage reference buffer mode.
0: Internal voltage reference mode disable (external voltage reference mode).
1: Internal voltage reference mode (reference buffer enable or hold mode) enable.
16.3.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04
Reset value: 0x0000 00XX

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

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

Bits 5:0 **TRIM[5:0]**: Trimming code
These bits are automatically initialized after reset with the trimming value stored in the Flash memory during the production test. Writing into these bits allows to tune the internal reference buffer voltage.

16.3.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Refer to *Section 2.2 on page 54* for the register boundary addresses.
17 Comparator (COMP)

17.1 Introduction
The STM32G071xx and STM32G081xx devices embed two ultra-low-power comparators, COMP1 and COMP2.
The comparators can be used for a variety of functions including:
- Wake-up from low-power mode triggered by an analog signal,
- Analog signal conditioning,
- Cycle-by-cycle current control loop when combined with a PWM output from a timer.

17.2 COMP main features
- Each comparator has configurable plus and minus inputs used for flexible voltage selection:
  - Multiplexed I/O pins
  - DAC Channel1 and Channel2
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
- Programmable hysteresis
- Programmable speed / consumption
- The outputs can be redirected to an I/O or to timer inputs for triggering:
  - Break events for fast PWM shutdowns
- Comparator outputs with blanking source
- The two comparators can be combined in a window comparator
- Each comparator has interrupt generation capability with wake-up from Sleep and Stop modes (through the EXTI controller)
17.3 COMP functional description

17.3.1 COMP block diagram

The block diagram of the comparators is shown in Figure 66: Comparator block diagram.

![Figure 66. Comparator block diagram](MSiv42190V1)

17.3.2 COMP pins and internal signals

The I/Os used as comparators inputs must be configured in analog mode in the GPIOs registers.

The comparator output can be connected to the I/Os using the alternate function channel given in “Alternate function mapping” table in the datasheet.

The output can also be internally redirected to a variety of timer input for the following purposes:

- Emergency shut-down of PWM signals, using BKin and BKin2 inputs
- Cycle-by-cycle current control, using OREF_CLR inputs
- Input capture for timing measures

It is possible to have the comparator output simultaneously redirected internally and externally.

<table>
<thead>
<tr>
<th>COMP1_INP</th>
<th>COMP1_INPSEL[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC5</td>
<td>00</td>
</tr>
<tr>
<td>PB2</td>
<td>01</td>
</tr>
<tr>
<td>PA1</td>
<td>10</td>
</tr>
<tr>
<td>Open</td>
<td>11</td>
</tr>
</tbody>
</table>
17.3.3 COMP reset and clocks

The COMP clock provided by the clock controller is synchronous with the APB2 clock.

There is no clock enable control bit provided in the RCC controller. Reset and clock enable bits are common for COMP and SYSCFG.

Table 81. COMP1 inverting input assignment

<table>
<thead>
<tr>
<th>COMP1_INM</th>
<th>COMP1_INMSEL[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>¼ V_REFINT</td>
<td>0000</td>
</tr>
<tr>
<td>½ V_REFINT</td>
<td>0001</td>
</tr>
<tr>
<td>¾ V_REFINT</td>
<td>0010</td>
</tr>
<tr>
<td>V_REFINT</td>
<td>0011</td>
</tr>
<tr>
<td>DAC Channel1</td>
<td>0100</td>
</tr>
<tr>
<td>DAC Channel2</td>
<td>0101</td>
</tr>
<tr>
<td>PB1</td>
<td>0110</td>
</tr>
<tr>
<td>PC4</td>
<td>0111</td>
</tr>
<tr>
<td>PA0</td>
<td>1000</td>
</tr>
<tr>
<td>¼ V_REFINT</td>
<td>&gt; 1000</td>
</tr>
</tbody>
</table>

Table 82. COMP2 non-inverting input assignment

<table>
<thead>
<tr>
<th>COMP2_INP</th>
<th>COMP2_INPSEL[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB4</td>
<td>00</td>
</tr>
<tr>
<td>PB6</td>
<td>01</td>
</tr>
<tr>
<td>PA3</td>
<td>10</td>
</tr>
<tr>
<td>Open</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 83. COMP2 inverting input assignment

<table>
<thead>
<tr>
<th>COMP2_INM</th>
<th>COMP2_INMSEL[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>¼ V_REFINT</td>
<td>0000</td>
</tr>
<tr>
<td>½ V_REFINT</td>
<td>0001</td>
</tr>
<tr>
<td>¾ V_REFINT</td>
<td>0010</td>
</tr>
<tr>
<td>V_REFINT</td>
<td>0011</td>
</tr>
<tr>
<td>DAC Channel1</td>
<td>0100</td>
</tr>
<tr>
<td>DAC Channel2</td>
<td>0101</td>
</tr>
<tr>
<td>PB3</td>
<td>0110</td>
</tr>
<tr>
<td>PB7</td>
<td>0111</td>
</tr>
<tr>
<td>PA2</td>
<td>1000</td>
</tr>
<tr>
<td>¼ V_REFINT</td>
<td>&gt; 1000</td>
</tr>
</tbody>
</table>
Note: **Important:** The polarity selection logic and the output redirection to the port works independently from the APB2 clock. This allows the comparator to work even in Stop mode.

### 17.3.4 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications having specific functional safety requirements, it is necessary to insure that the comparator programming cannot be altered in case of spurious register access or program counter corruption.

For this purpose, the comparator control and status registers can be write-protected (read-only).

Once the programming is completed, the COMPx LOCK bit can be set to 1. This causes the whole register to become read-only, including the COMPx LOCK bit.

The write protection can only be reset by a MCU reset.

### 17.3.5 Window comparator

The purpose of window comparator is to monitor the analog voltage if it is within specified voltage range defined by lower and upper threshold.

Two embedded comparators can be utilized to create window comparator. The monitored analog voltage is connected to the non-inverting (plus) inputs of comparators connected together and the upper and lower threshold voltages are connected to the inverting (minus) inputs of the comparators. Two non-inverting inputs can be connected internally together by enabling WINMODE bit to save one IO for other purposes.

#### Figure 67. Window mode

### 17.3.6 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed (for instance when exiting from low-power mode) to be able to force the hysteresis value using external components.
17.3.7 Comparator output blanking function

The purpose of the blanking function is to prevent the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches anti parallel diodes). It consists of a selection of a blanking window which is a timer output compare signal. The selection is done by software (refer to the comparator register description for possible blanking signals). Then, the complementary of the blanking signal is ANDed with the comparator output to provide the wanted comparator output. See the example provided in the figure below.
17.3.8 COMP power and speed modes

COMP1 and COMP2 power consumption versus propagation delay can be adjusted to have the optimum trade-off for a given application.

The bits PWRMODE[1:0] in COMPx_CSR registers can be programmed as follows:

- 00: High speed / full power
- 01, 10 or 11: Medium speed / medium power

17.4 COMP low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Low-power run</td>
<td>No effect.</td>
</tr>
<tr>
<td>Low-power sleep</td>
<td>No effect. COMP interrupts cause the device to exit the Low-power sleep mode.</td>
</tr>
<tr>
<td>Stop 0</td>
<td>No effect on the comparators.</td>
</tr>
<tr>
<td>Stop 1</td>
<td>Comparator interrupts cause the device to exit the Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The COMP registers are powered down and must be reinitialized after exiting Standby or Shutdown mode.</td>
</tr>
<tr>
<td>Shutdown</td>
<td></td>
</tr>
</tbody>
</table>

17.5 COMP interrupts

The comparator outputs are internally connected to the Extended interrupts and events controller. Each comparator has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit from low-power modes.

Refer to Interrupt and events section for more details.

To enable the COMPx interrupt, it is required to follow this sequence:

1. Configure and enable the EXTI line corresponding to the COMPx output event in interrupt mode and select the rising, falling or both edges sensitivity
2. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines
3. Enable the COMPx
17.6  COMP registers

17.6.1 Comparator 1 control and status register (COMP1_CSR)

COMP1_CSR is the comparator 1 control/status register. It contains all the bits / flags related to the comparator 1.

Address offset: 0x00

System reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>LOCK: COMP1_CSR register lock</td>
</tr>
<tr>
<td>30</td>
<td>VALUE: Comparator 1 output status</td>
</tr>
<tr>
<td>29:25</td>
<td>Reserved, must be kept at reset value</td>
</tr>
<tr>
<td>24:20</td>
<td>BLANKSEL[1:0]: Comparator 1 blanking source selector</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>LOCK: COMP1_CSR register lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>COMP1_CSR[31:0] register read/write bits can be written by software</td>
</tr>
<tr>
<td>1:</td>
<td>COMP1_CSR[31:0] register bits can be read but not written by software</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>VALUE: Comparator 1 output status</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit is read-only. It reflects the level of the comparator 1 output after the polarity selector and blanking, as indicated in Figure 66.</td>
<td></td>
</tr>
</tbody>
</table>

| Bits 29:25 | Reserved, must be kept at reset value |

<table>
<thead>
<tr>
<th>Bit 24:20</th>
<th>BLANKSEL[1:0]: Comparator 1 blanking source selector</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bitfield is controlled by software (if not locked). It selects the blanking source:</td>
<td></td>
</tr>
<tr>
<td>00000: None (no blanking)</td>
<td></td>
</tr>
<tr>
<td>xxx1: TIM1 OC4</td>
<td></td>
</tr>
<tr>
<td>xxx1x: TIM1 OC5</td>
<td></td>
</tr>
<tr>
<td>xx1xx: TIM2 OC3</td>
<td></td>
</tr>
<tr>
<td>x1xxx: TIM3 OC3</td>
<td></td>
</tr>
<tr>
<td>1xxxx: TIM15 OC2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop modes</th>
<th>Exit from Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMP1 output</td>
<td>VALUE in COMP1_CSR</td>
<td>through EXTI</td>
<td>yes</td>
<td>yes</td>
<td>N/A</td>
</tr>
<tr>
<td>COMP2 output</td>
<td>VALUE in COMP2_CSR</td>
<td>through EXTI</td>
<td>yes</td>
<td>yes</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Bit 19:18 **PWRMODE[1:0]**: Comparator 1 power mode selector
This bitfield is controlled by software (if not locked). It selects the power consumption and as a consequence the speed of the comparator 1:
00: High speed
01: Medium speed
others: Reserved

Bit 17:16 **HYST[1:0]**: Comparator 1 hysteresis selector
This bitfield is controlled by software (if not locked). It selects the hysteresis of the comparator 1:
00: None
01: Low
10: Medium
11: High

Bit 15 **POLARITY**: Comparator 1 polarity selector
This bit is controlled by software (if not locked). It selects the comparator 1 output polarity:
0: Non-inverted
1: Inverted

Bit 14 **WINOUT**: Comparator 1 output selector
This bit is controlled by software (if not locked). It selects the comparator 1 output:
0: COMP1_VALUE
1: COMP1_VALUE XOR COMP2_VALUE (required for window mode, see Figure 67)

Bits 13:12 Reserved, must be kept at reset value

Bit 11 **WINMODE**: Comparator 1 non-inverting input selector for window mode
This bit is controlled by software (if not locked). It selects the signal for COMP1_INP input of the comparator 1:
0: Signal selected with INPSEL[1:0] bitfield of this register
1: COMP2_INP signal of the comparator 2 (required for window mode, see Figure 67)

Bit 10 Reserved, must be kept at reset value

Bits 9:8 **INPSEL[1:0]**: Comparator 1 signal selector for non-inverting input
This bitfield is controlled by software (if not locked). It selects the signal for the non-inverting input COMP1_INP of the comparator 1 (also see the WINMODE bit):
00: PC5
01: PB2
10: PA1
11: None (open)
The COMP2_CSR is the comparator 2 control/status register. It contains all the bits/flags related to the comparator 2.

Address offset: 0x04

System reset value: 0x0000 0000

### Bits 7:4

**INMSEL[3:0]: Comparator 1 signal selector for inverting input INM**

This bitfield is controlled by software (if not locked). It selects the signal for the inverting input COMP1_INM of the comparator 1:

- 0000: 1/4 VREFINT
- 0001: 1/2 VREFINT
- 0010: 3/4 VREFINT
- 0011: VREFINT
- 0100: DAC channel 1
- 0101: DAC channel 2
- 0110: PB1
- 0111: PC4
- 1000: PA0
- > 1000: 1/4 VREFINT

### Bits 3:1

Reserved, must be kept at reset value

### Bit 0

**EN: Comparator 1 enable bit**

This bit is controlled by software (if not locked). It enables the comparator 1:

- 0: Disable
- 1: Enable

### 17.6.2 Comparator 2 control and status register (COMP2_CSR)

The COMP2_CSR is the comparator 2 control/status register. It contains all the bits/flags related to the comparator 2.

Address offset: 0x04

System reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td><strong>LOCK</strong>: COMP2_CSR register lock</td>
<td>rw</td>
<td>15</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td><strong>VALUE</strong>: Comparator 2 output status</td>
<td>rw</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Bit 31

**LOCK**: COMP2_CSR register lock

This bit is set by software and cleared by a system reset. It locks the whole content of the comparator 2 control register COMP2_CSR[31:0]:

- 0: COMP2_CSR[31:0] register read/write bits can be written by software
- 1: COMP2_CSR[31:0] register bits can be read but not written by software

### Bit 30

**VALUE**: Comparator 2 output status

This bit is read-only. It reflects the level of the comparator 2 output after the polarity selector and blanking, as indicated in Figure 66.

### Bits 29:25

Reserved, must be kept at reset value
Bit 24:20  **BLANKSEL[4:0]**: Comparator 2 blanking source selector  
This bitfield is controlled by software (if not locked). It selects the blanking source:  
00000: None (no blanking)  
xxxx1: TIM1 OC4  
xxx1x: TIM1 OC5  
xx1xx: TIM2 OC3  
x1xxx: TIM3 OC3  
1xxxx: TIM15 OC2

Bit 19:18  **PWRMODE[1:0]**: Comparator 2 power mode selector  
This bitfield is controlled by software (if not locked). It selects the power consumption and as a consequence the speed of the comparator 2:  
00: High speed  
01: Medium speed  
others: Reserved

Bit 17:16  **HYST[1:0]**: Comparator 2 hysteresis selector  
This bitfield is controlled by software (if not locked). It selects the hysteresis of the comparator 2:  
00: None  
01: Low  
10: Medium  
11: High

Bit 15  **POLARITY**: Comparator 2 polarity selector  
This bit is controlled by software (if not locked). It selects the comparator 2 output polarity:  
0: Non-inverted  
1: Inverted

Bit 14  **WINOUT**: Comparator 2 output selector  
This bit is controlled by software (if not locked). It selects the comparator 2 output:  
0: COMP2_VALUE  
1: COMP1_VALUE XOR COMP2_VALUE (required for window mode, see Figure 67)

Bits 13:12  Reserved, must be kept at reset value

Bit 11  **WINMODE**: Comparator 2 non-inverting input selector for window mode  
This bit is controlled by software (if not locked). It selects the signal for COMP2_INP input of the comparator 2:  
0: Signal selected with INPSEL[1:0] bitfield of this register  
1: COMP1_INP signal of the comparator 1 (required for window mode, see Figure 67)

Bit 10  Reserved, must be kept at reset value

Bits 9:8  **INPSEL[1:0]**: Comparator 2 signal selector for non-inverting input  
This bitfield is controlled by software (if not locked). It selects the signal for the non-inverting input COMP2_INP of the comparator 2 (also see the WINMODE bit):  
00: PB4  
01: PB6  
10: PA3  
11: None (open)
Bits 7:4 **INMSEL[3:0]**: Comparator 2 signal selector for inverting input INM

This bitfield is controlled by software (if not locked). It selects the signal for the inverting input COMP2_INM of the comparator 2:

- 0000: 1/4 VREFINT
- 0001: 1/2 VREFINT
- 0010: 3/4 VREFINT
- 0011: VREFINT
- 0100: DAC channel 1
- 0101: DAC channel 2
- 0110: PB3
- 0111: PB7
- 1000: PA2
- > 1000: 1/4 VREFINT

Bits 3:1 Reserved, must be kept at reset value

Bit 0 **EN**: Comparator 2 enable bit

This bit is controlled by software (if not locked). It enables the comparator 2:

- 0: Disable
- 1: Enable
### 17.6.3 COMP register map

The following table summarizes the comparator registers.

The comparator registers share SYSCFG peripheral register base addresses.

| Offset | Register   | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x200  | COMP1_CSR  | LOCK VALUE | BLANKSEL[4:0] | PWMODE[1:0] | HYST | POL | WINOUT | WINMODE | INPSEL[1:0] | INMSEL[3:0] | RES | RES | RES | RES | RES | RES | EN |
| Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x204  | COMP2_CSR  | LOCK VALUE | BLANKSEL[4:0] | PWMODE[1:0] | HYST | POL | WINOUT | WINMODE | INPSEL[1:0] | INMSEL[3:0] | RES | RES | RES | RES | RES | RES | EN |
| Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Refer to Section 2.2 on page 54 for the register boundary addresses.
18 True random number generator (RNG)

18.1 Introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG can be used to construct a NIST compliant Deterministic Random Bit Generator (DRBG), acting as a live entropy source.

The RNG true random number generator has been tested using German BSI statistical tests of AIS-31 (T0 to T8).

18.2 RNG main features

- The RNG delivers 32-bit true random numbers, produced by an analog entropy source processed by a high quality conditioning stage.
- It produces four 32-bit random samples every $16 \times \frac{f_{\text{AHB}}}{f_{\text{RNG}}}$ AHB clock cycles, if value is higher than 213 cycles (213 cycles otherwise).
- It allows embedded continuous basic health tests with associated error management – Includes too low sampling clock detection and repetition count tests.
- It can be disabled to reduce power consumption.
- It has an AMBA® AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).
18.3 **RNG functional description**

18.3.1 **RNG block diagram**

*Figure 70* shows the RNG block diagram.

![Figure 70. RNG block diagram](image)

18.3.2 **RNG internal signals**

*Table 87* describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rng_it</td>
<td>Digital output</td>
<td>RNG global interrupt request</td>
</tr>
<tr>
<td>rng_hclk</td>
<td>Digital input</td>
<td>AHB clock</td>
</tr>
<tr>
<td>rng_clk</td>
<td>Digital input</td>
<td>RNG dedicated clock, asynchronous to rng_hclk</td>
</tr>
</tbody>
</table>
18.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals. The RNG implements the entropy source model pictured on Figure 71, and provides three main functions to the application:

- Collects the bitstring output of the entropy source box
- Obtains samples of the noise source for validation purpose
- Collects error messages from continuous health tests

Figure 71. Entropy source model

The main components of the RNG are:

- A source of physical randomness (analog noise sources)
- A digitization stage for those analog noise sources
- A stage delivering a cryptographically conditioned noise source
- Output buffers, for both entropy source output (buffered) and noise source samples (also buffered)
- A health monitoring block performing tests on the whole entropy source

All those components are detailed below.

Noise source

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. It is composed of:

- Two analog noise sources, each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in Section 18.3.8: RNG low-power usage.
- A sampling stage of these outputs clocked by a dedicated clock input (rng_clk), delivering a 2-bit raw data output.
This noise source sampling is independent to the AHB interface clock frequency (\texttt{rng\_hclk}).

**Note:** In Section 18.6: RNG entropy source validation recommended RNG clock frequencies are given.

**Post processing**

The sample values obtained from a true random noise source consist of 2-bit bitstrings. Because this noise source output is biased, the RNG implements a post-processing component that reduces that bias to a tolerable level.

More specifically, for each of the two noise source bits the RNG takes half of the bits from the sampled noise source, and half of the bits from inverted sampled noise source. Thus, if the source generates more ‘1’ than ‘0’ (or the opposite), it is filtered.

**Conditioning**

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit).

**Note:** The latency during the RNG initialization is described in Section 18.5: RNG processing time.

Also note that post-processing computations are triggered when at least 32 bits of raw data has been received and when output FIFO needs a refill. Thus the RNG output entropy is maximum when the RNG 128-bit FIFO is emptied by application after 64 RNG clock cycles.

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in Section 18.5: RNG processing time.

The conditioning component is clocked by the faster AHB clock.

**Output buffer**

A data output buffer can store up to four 32-bit words which have been output from the conditioning component. When four words have been read from the output FIFO through the RNG\_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register 213 AHB clock cycles later.

Whenever a random number is available through the RNG\_DR register the DRDY flag transitions from “0” to “1”. This flag remains high until output buffer becomes empty after reading four words from the RNG\_DR register.

**Note:** When interrupts are enabled an interrupt is generated when this data ready flag transitions from “0” to “1”. Interrupt is then cleared automatically by the RNG as explained above.
Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features to the value recommended for register: RNG_HCTR (in Section 18.6.2).

1. the Continuous health tests, running indefinitely on the output of the noise source
   a) One of the noise source has provided more than 64 consecutive bits at a constant value ("0" or "1"), or more than 32 consecutive occurrence of two bits patterns ("01" or "10")
   b) Both noise sources have delivered more than 32 consecutive bits at a constant value ("0" or "1"), or more than 16 consecutive occurrence of two bits patterns ("01" or "10")

2. Vendor specific continuous test
   a) Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle is smaller than AHB clock cycle divided by 32.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in Section 18.3.7: Error management.

Note: An interrupt can be generated when an error is detected.

18.3.4 RNG initialization

The RNG simplified state machine is pictured on Figure 72.

After enabling the RNG (RNGEN=1 in RNG_CR) the following chain of events occurs:

1. The analog noise source is enabled, and logic immediately starts sampling the analog output, filling 128-bit conditioning shift register

2. The conditioning logic is enabled and post-processing context is initialized using two 128 noise source bits.

3. The conditioning stage internal input data buffer is filled again with 128-bit and one conditioning round is performed. The output buffer is then filled with post processing result.

4. The output buffer is refilled automatically according to the RNG usage.
The associated initialization time can be found in Section 18.5: RNG processing time.

Figure 72. RNG initialization overview

18.3.5 RNG operation

Normal operations

To run the RNG using interrupts the following steps are recommended:

1. Enable the interrupts by setting the IE bit in the RNG_CR register. At the same time enable the RNG by setting the bit RNGEN=1.

2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits should be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data is available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit will still be high). If one or both of above conditions are false, the RNG_DR register must not be read. If an error occurred error recovery sequence described in Section 18.3.7 shall be used.
To run the RNG in polling mode following steps are recommended:

1. Enable the random number generation by setting the RNGEN bit to “1” in the RNG_CR register.
2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits should be set to 0)
   - A random number is ready (the DRDY bit should be set to 1)
3. If above conditions are true read the content of the RNG_DR register up to four consecutive times. If valid data is available in the conditioning output buffer four additional words can be read by the application (in this case the DRDY bit will still be high). If one or both of above conditions are false, the RNG_DR register must not be read. If an error occurred error recovery sequence described in Section 18.3.7 shall be used.

Note: When data is not ready (DRDY=“0”) RNG_DR returns zero.

Low-power operations

If the power consumption is a concern to the application, low-power strategies can be used, as described in Section 18.3.8: RNG low-power usage.

Software post-processing

No specific software post-processing/conditioning is required to meet AIS-31 approvals. If a NIST approved DRBG with 128 bits of security strength is required an approved random generator software must be built around the RNG true random number generator.

Built-in health check functions are described in Section 18.3.3: Random number generation.

18.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock is used for noise source sampling. Recommended clock configurations are detailed in Section 18.6: RNG entropy source validation.

Note: When the CED bit in the RNG_CR register is set to “0”, the RNG clock frequency should be higher than AHB clock frequency divided by 32, otherwise the clock checker will always flag a clock error (CECS=1 in the RNG_SR register).

See Section 18.3.1: RNG block diagram for details (AHB and RNG clock domains).

18.3.7 Error management

In parallel to random number generation an health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to “1” both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application should check that the RNG clock is configured correctly (see Section 18.3.6: RNG clocking) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when clocking condition is normal.


**Note:** The clock error has no impact on generated random numbers, i.e. application can still read RNG_DR register.

CEIS is set only when CECS is set to “1” by RNG.

### Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to “1” both **SEIS** and **SECS** bits to indicate that a seed error occurred. If a value is available in the RNG_DR register, it must not be used as it may not have enough entropy. If the error was detected during the initialization phase the whole initialization sequence will be automatically restarted by the RNG.

The following sequence shall be used to fully recover from a seed error after the RNG initialization:

1. Clear the SEIS bit by writing it to “0”.
2. Read out 12 words from the RNG_DR register, and discard each of them in order to clean the pipeline.
3. Confirm that SEIS is still cleared. Random number generation is back to normal.

### 18.3.8 RNG low-power usage

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to “1” by setting the RNGEN bit to “0” in the RNG_CR register. As the post-processing logic and the output buffer remain operational while RNGEN=’0’ following features are available to software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG_DR register. If it is not the case the RNG must be re-enabled by the application until at least 32 new bits have been collected from the noise source and a complete conditioning round has been done. It corresponds to 16 RNG clock cycles to sample new bits, and 216 AHB clock cycles to run a conditioning round.

When disabling the RNG the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (i.e. well before the DRDY bit rises for the first time), the initialization sequence will resume from where it was stopped when RNGEN bit is set to “1”.

### 18.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see Section 18.3.7: Error management
- Clock error, see Section 18.3.7: Error management

Dedicated interrupt enable control bits are available as shown in Table 88.
The user can enable or disable the above interrupt sources individually by changing the mask bits or the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

*Note:* Interrupts are generated only when RNG is enabled.

### 18.5 RNG processing time

The conditioning stage can produce four 32-bit random numbers every \(16 \times \frac{f_{\text{AHB}}}{f_{\text{RNG}}}\) clock cycles, if the value is higher than 213 cycles (213 cycles otherwise). More time is needed for the first set of random numbers after the device exits reset (see *Section 18.3.4: RNG initialization*). Indeed, after enabling the RNG for the first time, random data is first available after either:

- 128 RNG clock cycles + 426 AHB cycles, if \(f_{\text{AHB}} < f_{\text{threshold}}\)
- 192 RNG clock cycles + 213 AHB cycles, if \(f_{\text{AHB}} \geq f_{\text{threshold}}\)

With \(f_{\text{threshold}} = \frac{(213 \times f_{\text{RNG}})}{64}\)

### 18.6 RNG entropy source validation

#### 18.6.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using German BSI AIS-31 statistical tests (T0 to T8). The results can be provided on demand or the customer can reproduce the tests.

#### 18.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock \(\text{rng}_{\text{clk}} = 48\) MHz (CED bit = ‘0’ in RNG_CR register) and \(\text{rng}_{\text{clk}} = 400\) kHz (CED bit = ‘1’ in RNG_CR register).

#### 18.6.3 Data collection

In order to run statistical tests it is required to collect samples from the entropy source at raw data level as well as at the output of the entropy source.

Contact STMicroelectronics if above samples need to be retrieved for your product.

---

**Table 88. RNG interrupt requests**

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNG</td>
<td>Data ready flag</td>
<td>DRDY</td>
<td>IE</td>
<td>None (automatic)</td>
</tr>
<tr>
<td></td>
<td>Seed error flag</td>
<td>SEIS</td>
<td>IE</td>
<td>Write 0 to SEIS or write CONDRST to 1 then to 0</td>
</tr>
<tr>
<td></td>
<td>Clock error flag</td>
<td>CEIS</td>
<td>IE</td>
<td>Write 0 to CEIS</td>
</tr>
</tbody>
</table>
18.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

18.7.1 RNG control register (RNG_CR)

Address offset: 0x000
Reset value: 0x0000 0000

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

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

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

Bit 5 **CED**: Clock error detection

0: Clock error detection is enable
1: Clock error detection is disable

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, i.e. to enable or disable CED the RNG must be disabled.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE**: Interrupt Enable

0: RNG Interrupt is disabled
1: RNG Interrupt is enabled. An interrupt is pending as soon as DRDY='1', SEIS='1' or CEIS=1 in the RNG_SR register.

Bit 2 **RNGEN**: True random number generator enable

0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.
1: True random number generator is enabled.

Bits 1:0 Reserved, must be kept at reset value.
### 18.7.2 RNG status register (RNG_SR)

Address offset: 0x004  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w0</td>
<td>rc_w0</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

#### Bit 6 SEIS: Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0. Writing 1 has no effect.

- 0: No faulty sequence detected
- 1: At least one faulty sequence has been detected. See SECS bit description for details.  
An interrupt is pending if IE = 1 in the RNG_CR register.

#### Bit 5 CEIS: Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

- 0: The RNG clock is correct (fRNGCLK > fHCLK/32)
- 1: The RNG has been detected too slow (fRNGCLK < fHCLK/32)

An interrupt is pending if IE = 1 in the RNG_CR register.

Bits 4:3 Reserved, must be kept at reset value.

#### Bit 2 SECS: Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequence has been detected:

- One of the noise source has provided more than 64 consecutive bits at a constant value ("0" or "1"), or more than 32 consecutive occurrence of two bit patterns ("01" or "10")
- Both noise sources have delivered more than 32 consecutive bits at a constant value ("0" or "1"), or more than 16 consecutive occurrence of two bit patterns ("01" or "10")

#### Bit 1 CECS: Clock error current status

0: The RNG clock is correct (fRNGCLK > fHCLK/32). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow (fRNGCLK < fHCLK/32).

*Note:* CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

#### Bit 0 DRDY: Data Ready

0: The RNG_DR register is not yet valid, no random data is available.

1: The RNG_DR register contains valid random data.

Once the output buffer becomes empty (after reading the RNG_DR register), this bit returns to 0 until a new random value is generated.

*Note:* The DRDY bit can rise when the peripheral is disabled (RNGEN=0 in the RNG_CR register).

If IE=1 in the RNG_CR register, an interrupt is generated when DRDY=1.
18.7.3 **RNG data register (RNG_DR)**

Address offset: 0x008  
Reset value: 0x0000 0000  

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. After being read this register delivers a new random value after 216 periods of AHB clock if the output FIFO is empty.

The content of this register is valid when DRDY=1, even if RNGEN=0.

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

Bits 31:0 **RNDATA[31:0]:** Random data  
32-bit random data which are valid when DRDY=1. When DRDY=0 RNDATA value is zero.
18.7.4 RNG register map

Table 89 gives the RNG register map and reset values.

Table 89. RNG register map and reset map

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | RNG_CR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x004  | RNG_SR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | RNG_DR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | RNDATA[31:0]  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
19 AES hardware accelerator (AES)

19.1 Introduction

The AES hardware accelerator (AES) encrypts or decrypts data, using an algorithm and implementation fully compliant with the advanced encryption standard (AES) defined in Federal information processing standards (FIPS) publication 197.

Multiple chaining modes are supported (ECB, CBC, CTR, GCM, GMAC, CCM), for key sizes of 128 or 256 bits.

The AES accelerator is a 32-bit AHB peripheral. It supports DMA single transfers for incoming and outgoing data (two DMA channels required).

The AES peripheral provides hardware acceleration to AES cryptographic algorithms packaged in STM32 cryptographic library.

AES is an AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated). Other than 32-bit writes may corrupt the register content.

19.2 AES main features

- Compliance with NIST “Advanced encryption standard (AES), FIPS publication 197” from November 2001
- 128-bit data block processing
- Support for cipher key lengths of 128-bit and 256-bit
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 51 or 75 clock cycle latency in ECB mode for processing one 128-bit block of data with, respectively, 128-bit or 256-bit key
- Integrated key scheduler with its key derivation stage (ECB or CBC decryption only)
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only
- 256-bit register for storing the cryptographic key (eight 32-bit registers)
- 128-bit register for storing initialization vector (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control with support of single-transfer direct memory access (DMA) using two channels (one for incoming data, one for processed data)
- Data-swapping logic to support 1-, 8-, 16- or 32-bit data
- Possibility for software to suspend a message if AES needs to process another message with a higher priority, then resume the original message
19.3 **AES implementation**

The devices have a single instance of AES peripheral.

19.4 **AES functional description**

19.4.1 **AES block diagram**

*Figure 73* shows the block diagram of AES.

![Figure 73. AES block diagram](image_url)

19.4.2 **AES internal signals**

*Table 90* describes the user relevant internal signals interfacing the AES peripheral.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aes_hclk</td>
<td>digital input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>aes_it</td>
<td>digital output</td>
<td>AES interrupt request</td>
</tr>
<tr>
<td>aes_in_dma</td>
<td>digital input/output</td>
<td>Input DMA single request/acknowledge</td>
</tr>
<tr>
<td>aes_out_dma</td>
<td>digital input/output</td>
<td>Output DMA single request/acknowledge</td>
</tr>
</tbody>
</table>
19.4.3 AES cryptographic core

Overview

The AES cryptographic core consists of the following components:
- AES algorithm (AEA)
- multiplier over a binary Galois field (GF2mul)
- key input
- initialization vector (IV) input
- chaining algorithm logic (XOR, feedback/counter, mask)

The AES core works on 128-bit data blocks (four words) with 128-bit or 256-bit key length. Depending on the chaining mode, the AES requires zero or one 96-bit initialization vector IV (and a 32-bit counter field).

The AES features the following modes of operation:
- **Mode 1:** Plaintext encryption using a key stored in the AES_KEYRx registers
- **Mode 2:** ECB or CBC decryption key preparation. It must be used prior to selecting Mode 3 with ECB or CBC chaining modes. The key prepared for decryption is stored automatically in the AES_KEYRx registers. Now the AES peripheral is ready to switch to Mode 3 for executing data decryption.
- **Mode 3:** Ciphertext decryption using a key stored in the AES_KEYRx registers. When ECB and CBC chaining modes are selected, the key must be prepared beforehand, through Mode 2.
- **Mode 4:** ECB or CBC ciphertext single decryption using the key stored in the AES_KEYRx registers (the initial key is derived automatically).

Note: Mode 2 and mode 4 are only used when performing ECB and CBC decryption.

When Mode 4 is selected only one decryption can be done, therefore usage of Mode 2 and Mode 3 is recommended instead.

The operating mode is selected by programming the MODE[1:0] bitfield of the AES_CR register. It may be done only when the AES peripheral is disabled.

Typical data processing

Typical usage of the AES is described in Section 19.4.4: AES procedure to perform a cipher operation on page 436.

Note: The outputs of the intermediate AEA stages are never revealed outside the cryptographic boundary, with the exclusion of the IVI bitfield.
Chaining modes

The following chaining modes are supported by AES, selected through the CHMOD[2:0] bitfield of the AES_CR register:
- Electronic code book (ECB)
- Cipher block chaining (CBC)
- Counter (CTR)
- Galois counter mode (GCM)
- Galois message authentication code (GMAC)
- Counter with CBC-MAC (CCM)

Note: The chaining mode may be changed only when AES is disabled (bit EN of the AES_CR register cleared).

Principle of each AES chaining mode is provided in the following subsections.
Detailed information is in dedicated sections, starting from Section 19.4.8: AES basic chaining modes (ECB, CBC).

Electronic codebook (ECB) mode

Figure 74. ECB encryption and decryption principle

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

Note: For decryption, a special key scheduling is required before processing the first block.
Cipher block chaining (CBC) mode

Figure 75. CBC encryption and decryption principle

In CBC mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

**Note:** For decryption, a special key scheduling is required before processing the first block.
Counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XORed with the plaintext to obtain the ciphertext as specified in NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.

Note: Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption, since in this chaining scheme the AES core is always used in encryption mode for producing the key stream, or counter blocks.
Galois/counter mode (GCM)

In Galois/counter mode (GCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. It requires an initial value and a particular 128-bit block at the end of the message.

Galois message authentication code (GMAC) principle

Galois message authentication code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.
GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload).

**Counter with CBC-MAC (CCM) principle**

In Counter with cipher block chaining-message authentication code (CCM) mode, the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and the corresponding MAC (also known as tag). It is described by NIST in *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

CCM mode is based on AES in counter mode for confidentiality and it uses CBC for computing the message authentication code. It requires an initial value.

Like GCM, the CCM chaining mode can be applied on a message composed only by plaintext authenticated data (that is, only header, no payload). Note that this way of using CCM is not called CMAC (it is not similar to GCM/GMAC), and its usage is not recommended by NIST.

**19.4.4 AES procedure to perform a cipher operation**

**Introduction**

A typical cipher operation is explained below. Detailed information is provided in sections starting from *Section 19.4.8: AES basic chaining modes (ECB, CBC)*.

The flowcharts shown in *Figure 80* and *Figure 81* describe the way STM32 cryptographic library implements the AES algorithm. AES accelerates the execution of the AES-128 and AES-256 cryptographic algorithms in ECB, CBC, CTR, CCM, and GCM operating modes.

Figure 80. STM32 cryptolib AES flowchart examples

**Encryption**
- Begin
- AES_x encrypt init
  - Error status
    - success
    - AES_x encrypt
      - Error status
        - success
        - AES_x encrypt finish/ final
          - Error status
            - success
            - End

**Decryption**
- Begin
- AES_x decrypt init
  - Error status
    - success
    - AES_x decrypt
      - Error status
        - success
        - AES_x decrypt finish/ final
          - Error status
            - success
            - End

Error status
Data to append
Initialization of AES

To initialize AES, first disable it by clearing the EN bit of the AES_CR register. Then perform the following steps in any order:

- Configure the AES mode, by programming the MODE[1:0] bitfield of the AES_CR register.
  - For encryption, Mode 1 must be selected (MODE[1:0] = 00).
  - For decryption, Mode 3 must be selected (MODE[1:0] = 10), unless ECB or CBC chaining modes are used. In this latter case, an initial key derivation of the encryption key must be performed, as described in Section 19.4.5: AES decryption key preparation.

- Select the chaining mode, by programming the CHMOD[2:0] bitfield of the AES_CR register.

- Configure the key size (128-bit or 256-bit), with the KEYSIZE bitfield of the AES_CR register.

- Write a symmetric key into the AES_KEYRx registers (4 or 8 registers depending on the key size).

- Configure the data type (1-, 8-, 16- or 32-bit), with the DATATYPE[1:0] bitfield in the AES_CR register.

- When it is required (for example in CBC or CTR chaining modes), write the initialization vector into the AES_IVRx registers.
Data append

This section describes different ways of appending data for processing, where the size of data to process is not a multiple of 128 bits.

For ECB or CBC mode, refer to Section 19.4.6: AES ciphertext stealing and data padding. The last block management in these cases is more complex than in the sequence described in this section.

Data append through polling

This method uses flag polling to control the data append through the following sequence:

1. Enable the AES peripheral by setting the EN bit of the AES_CR register.
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the AES_DINR register.
   b) Wait until the status flag CCF is set in the AES_SR, then read the four data words from the AES_DOUTR register.
   c) Clear the CCF flag, by setting the CCFC bit of the AES_CR register.
   d) If the data block just processed is the second-last block of the message and the significant data in the last block to process is inferior to 128 bits, pad the remainder of the last block with zeros and, in case of GCM payload encryption or CCM payload decryption, specify the number of non-valid bytes, using the NPBLB bitfield of the AES_CR register, for AES to compute a correct tag.
3. Discard the data that is not part of the payload, then disable the AES peripheral by clearing the EN bit of the AES_CR register.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the AES_DINR register, to allow sending the key to the AES processor.

NPBLB bits are not used in header phase of GCM, GMAC and CCM chaining modes.

Data append using interrupt

The method uses interrupt from the AES peripheral to control the data append, through the following sequence:

1. Enable interrupts from AES by setting the CCFIE bit of the AES_CR register.
2. Enable the AES peripheral by setting the EN bit of the AES_CR register.
3. Write first four input data words into the AES_DINR register.
4. Handle the data in the AES interrupt service routine, upon interrupt:
   a) Read four output data words from the AES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCFC bit of the AES_CR register.
   c) If the data block just processed is the second-last block of an message and the significant data in the last block to process is inferior to 128 bits, pad the remainder of the last block with zeros and, in case of GCM payload encryption or CCM payload decryption, specify the number of non-valid bytes, using the NPBLB
bitfield of the AES_CR register, for AES to compute a correct tag. Then proceed
with point 4e).

d) If the data block just processed is the last block of the message, discard the data
that is not part of the payload, then disable the AES peripheral by clearing the EN
bit of the AES_CR register and quit the interrupt service routine.

e) Write next four input data words into the AES_DINR register and quit the interrupt
service routine.

Note: AES is tolerant of delays between consecutive read or write operations, which allows, for
example, an interrupt from another peripheral to be served between two AES computations.
NPBLB bits are not used in header phase of GCM, GMAC and CCM chaining modes.

Data append using DMA

With this method, all the transfers and processing are managed by DMA and AES. To use
the method, proceed as follows:

1. Prepare the last four-word data block (if the data to process does not fill it completely),
   by padding the remainder of the block with zeros.

2. Configure the DMA controller so as to transfer the data to process from the memory to
   the AES peripheral input and the processed data from the AES peripheral output to the
   memory, as described in Section 19.4.16: AES DMA interface. Configure the DMA
   controller so as to generate an interrupt on transfer completion. In case of GCM
   payload encryption or CCM payload decryption, DMA transfer must not include
   the last four-word block if padded with zeros. The sequence described in Data append
   through polling must be used instead for this last block, because NPBLB bits must be
   setup before processing the block, for AES to compute a correct tag.

3. Enable the AES peripheral by setting the EN bit of the AES_CR register

4. Enable DMA requests by setting the DMAINEN and DMAOUTEN bits of the AES_CR
   register.

5. Upon DMA interrupt indicating the transfer completion, get the AES-processed data
   from the memory.

Note: The CCF flag has no use with this method, because the reading of the AES_DOUTR
register is managed by DMA automatically, without any software action, at the end of the
computation phase.
NPBLB bits are not used in header phase GCM, GMAC and CCM chaining modes.

19.4.5 AES decryption key preparation

For an ECB or CBC decryption, a key for the first round of decryption must be derived from
the key of the last round of encryption. This is why a complete key schedule of encryption is
required before performing the decryption. This key preparation is not required for AES
decryption in modes other than ECB or CBC.

Recommended method is to select the Mode 2 by setting to 01 the MODE[1:0] bitfield of the
AES_CR (key process only), then proceed with the decryption by setting MODE[1:0] to 10
(Mode 3, decryption only). Mode 2 usage is described below:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.

2. Select Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES_CR. The
   CHMOD[2:0] bitfield is not significant in this case because this key derivation mode is
   independent of the chaining algorithm selected.

3. Set key length to 128 or 256 bits, via KEYSIZE bit of AES_CR register.
4. Write the AES_KEYRx registers (128 or 256 bits) with encryption key, as shown in Figure 82. Writes to the AES_IVRx registers have no effect.

5. Enable the AES peripheral, by setting the EN bit of the AES_CR register.

6. Wait until the CCF flag is set in the AES_SR register.

7. Clear the CCF flag. Derived key is available in AES core, ready to use for decryption. Application can also read the AES_KEYRx register to obtain the derived key if needed, as shown in Figure 82 (the processed key is loaded automatically into the AES_KEYRx registers).

Note: The AES is disabled by hardware when the derivation key is available.
To restart a derivation key computation, repeat steps 4, 5, 6, and 7.

Figure 82. Encryption key derivation for ECB/CBC decryption (Mode 2)

If the software stores the initial key prepared for decryption, it is enough to do the key schedule operation only once for all the data to be decrypted with a given cipher key.

Note: The operation of the key preparation lasts 59 or 82 clock cycles, depending on the key size (128- or 256-bit).

Note: Alternative key preparation is to select Mode 4 by setting to 11 the MODE[1:0] bitfield of the AES_CR register. In this case Mode 3 cannot be used.

19.4.6 AES ciphertext stealing and data padding

When using AES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (128 bits), ciphertext stealing techniques are used, such as those described in NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode. Since the AES peripheral does not support such techniques, the application must complete the last block of input data using data from the second last block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, when AES is used in other modes than ECB or CBC, an incomplete input data block (that is, block with input data shorter than 128 bits) must be padded with zeros prior to encryption (that is, extra bits must be appended to the trailing end of the data string). After decryption, the extra bits must be discarded. As AES does not implement automatic data padding operation to the last block, the application must follow the recommendation given in Section 19.4.4: AES procedure to perform a cipher operation on page 436 to manage messages the size of which is not a multiple of 128 bits.
Note: Padding data are swapped in a similar way as normal data, according to the DATATYPE[1:0] field of the AES_CR register (see Section 19.4.13: AES data registers and data swapping on page 463 for details).

19.4.7 AES task suspend and resume

A message can be suspended if another message with a higher priority must be processed. When this highest priority message is sent, the suspended message can resume in both encryption or decryption mode.

Suspend/resume operations do not break the chaining operation and the message processing can resume as soon as AES is enabled again to receive the next data block. Figure 83 gives an example of suspend/resume operation: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 83. Example of suspend mode management

A detailed description of suspend/resume operations is in the sections dedicated to each AES mode.

19.4.8 AES basic chaining modes (ECB, CBC)

Overview

This section gives a brief explanation of the four basic operation modes provided by the AES computing core: ECB encryption, ECB decryption, CBC encryption and CBC decryption. For detailed information, refer to the FIPS publication 197 from November 26, 2001.
Figure 84 illustrates the electronic codebook (ECB) encryption.

**Figure 84. ECB encryption**

In ECB encrypt mode, the 128-bit plaintext input data block $P_x$ in the AES_DINR register first goes through bit/byte/half-word swapping. The swap result $I_x$ is processed with the AES core set in encrypt mode, using a 128- or 256-bit key. The encryption result $O_x$ goes through bit/byte/half-word swapping, then is stored in the AES_DOUTR register as 128-bit ciphertext output data block $C_x$. The ECB encryption continues in this way until the last complete plaintext block is encrypted.

Figure 85 illustrates the electronic codebook (ECB) decryption.

**Figure 85. ECB decryption**

To perform an AES decryption in the ECB mode, the secret key has to be prepared by collecting the last-round encryption key (which requires to first execute the complete key schedule for encryption), and using it as the first-round key for the decryption of the ciphertext. This preparation is supported by the AES core.

In ECB decrypt mode, the 128-bit ciphertext input data block $C_1$ in the AES_DINR register first goes through bit/byte/half-word swapping. The keying sequence is reversed compared to that of the ECB encryption. The swap result $I_1$ is processed with the AES core set in decrypt mode, using the formerly prepared decryption key. The decryption result goes through bit/byte/half-word swapping, then is stored in the AES_DOUTR register as 128-bit plaintext output data block $P_1$. The ECB decryption continues in this way until the last complete ciphertext block is decrypted.
Figure 86 illustrates the cipher block chaining (CBC) encryption mode.

In CBC encrypt mode, the first plaintext input block, after bit/byte/half-word swapping (P1'), is XOR-ed with a 128-bit IVI bitfield (initialization vector and counter), producing the I1 input data for encrypt with the AES core, using a 128- or 256-bit key. The resulting 128-bit output block O1, after swapping operation, is used as ciphertext C1. The O1 data is then XOR-ed with the second-block plaintext data P2' to produce the I2 input data for the AES core to produce the second block of ciphertext data. The chaining of data blocks continues in this way until the last plaintext block in the message is encrypted.

If the message size is not a multiple of 128 bits, the final partial data block is encrypted in the way explained in Section 19.4.6: AES ciphertext stealing and data padding.

Figure 87 illustrates the cipher block chaining (CBC) decryption.

In CBC decrypt mode, like in ECB decrypt mode, the secret key must be prepared to perform an AES decryption.

After the key preparation process, the decryption goes as follows: the first 128-bit ciphertext block (after the swap operation) is used directly as the AES core input block I1 for decrypt operation, using the 128-bit or 256-bit key. Its output O1 is XOR-ed with the 128-bit IVI field (that must be identical to that used during encryption) to produce the first plaintext block P1.
The second ciphertext block is processed in the same way as the first block, except that the I1 data from the first block is used in place of the initialization vector.

The decryption continues in this way until the last complete ciphertext block is decrypted. If the message size is not a multiple of 128 bits, the final partial data block is decrypted in the way explained in Section 19.4.6: AES ciphertext stealing and data padding.

For more information on data swapping, refer to Section 19.4.13: AES data registers and data swapping.

**ECB/CBC encryption sequence**

The sequence of events to perform an ECB/CBC encryption (more detail in Section 19.4.4):

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select the Mode 1 by setting to 00 the MODE[1:0] bitfield of the AES_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES_CR register to 000 or 001, respectively. Data type can also be defined, using DATATYPE[1:0] bitfield.
3. Select 128- or 256-bit key length through the KEYSIZE bit of the AES_CR register.
4. Write the AES_KEYRx registers (128 or 256 bits) with encryption key. Fill the AES_IVRx registers with the initialization vector data if CBC mode has been selected.
5. Enable the AES peripheral by setting the EN bit of the AES_CR register.
6. Write the AES_DINR register four times to input the plaintext (MSB first), as shown in Figure 88.
7. Wait until the CCF flag is set in the AES_SR register.
8. Read the AES_DOUTR register four times to get the ciphertext (MSB first) as shown in Figure 88. Then clear the CCF flag by setting the CCFC bit of the AES_CR register.
9. Repeat steps 6-7-8 to process all the blocks with the same encryption key.

**Figure 88. ECB/CBC encryption (Mode 1)**

<table>
<thead>
<tr>
<th>MSB</th>
<th>WR PT3</th>
<th>WR PT2</th>
<th>WR PT1</th>
<th>WR PT0</th>
<th>Wait until flag CCF = 1</th>
<th>RD CT3</th>
<th>RD CT2</th>
<th>RD CT1</th>
<th>RD CT0</th>
<th>LSB</th>
</tr>
</thead>
</table>

Input phase
4 write operations into AES_DINR[31:0]

Computation phase

Output phase
4 read operations of AES_DOUTR[31:0]

**ECB/CBC decryption sequence**

The sequence of events to perform an AES ECB/CBC decryption is as follows (more detail in Section 19.4.4):

1. Follow the steps described in Section 19.4.5: AES decryption key preparation on page 440, in order to prepare the decryption key in AES core.
2. Select the Mode 3 by setting to 10 the MODE[1:0] bitfield of the AES_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES_CR register to 001.
3. Write the AES_KEYRx registers (128 or 256 bits) with decryption key.
4. Write the AES_IVRx registers with the initialization vector data if CBC mode has been selected.
5. Enable the AES peripheral by setting the EN bit of the AES_CR register.
6. Write the AES_DINR register four times to input the ciphertext (MSB first), as shown in Figure 88.
7. Wait until the CCF flag is set in the AES_SR register.
8. Read the AES_DOUTR register four times to get the plaintext (MSB first) as shown in Figure 88. Then clear the CCF flag by setting the CCFC bit of the AES_CR register.
9. Repeat steps 6-7-8 to process all the blocks with the same decryption key.
register to 000 or 001, respectively. Data type can also be defined, using DATATYPE\[1:0\] bitfield.

3. Select key length of 128 or 256 bits via KEYSIZE bitfield of the AES_CR register.

4. Write the AES_IVRx registers with the initialization vector (required in CBC mode only).

5. Enable AES by setting the EN bit of the AES_CR register.

6. Write the AES_DINR register four times to input the cipher text (MSB first), as shown in Figure 89.

7. Wait until the CCF flag is set in the AES_SR register.

8. Read the AES_DOUTR register four times to get the plain text (MSB first), as shown in Figure 89. Then clear the CCF flag by setting the CCFC bit of the AES_CR register.

9. Repeat steps 6-7-8 to process all the blocks encrypted with the same key.

Figure 89. ECB/CBC decryption (Mode 3)

Suspend/resume operations in ECB/CBC modes

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES_CR register.

2. If DMA is not used, read four times the AES_DOUTR register to save the last processed block. If DMA is used, wait until the CCF flag is set in the AES_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES_CR register.

3. If DMA is not used, poll the CCF flag of the AES_SR register until it becomes 1 (computation completed).

4. Clear the CCF flag by setting the CCFC bit of the AES_CR register.

5. Save initialization vector registers (only required in CBC mode as AES_IVRx registers are altered during the data processing).

6. Disable the AES peripheral by clearing the bit EN of the AES_CR register.

7. Save the current AES configuration in the memory (except AES initialization vector values). Clear the key registers if they are not needed to process the higher priority message.

8. If DMA is used, save the DMA controller status (pointers for IN and OUT data transfers, number of remaining bytes, and so on).
Note: In point 7, the derived key information stored in AES_KEYRx registers can optionally be saved in memory if the interrupted process is a decryption. Otherwise those registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:
1. If DMA is used, configure the DMA controller so as to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Ensure that AES is disabled (the EN bit of the AES_CR must be 0).
3. Restore the AES_CR and AES_KEYRx register setting, using the values of the saved configuration. In case of decryption, derived key information can be written in AES_KEYRx register instead of the original key value.
4. Prepare the decryption key as described in Section 19.4.5: AES decryption key preparation (only required for ECB or CBC decryption). This step is not necessary if derived key information has been loaded in AES_KEYRx registers.
5. Restore AES_IVRx registers using the saved configuration (only required in CBC mode).
6. Enable the AES peripheral by setting the EN bit of the AES_CR register.
7. If DMA is used, enable AES DMA transfers by setting the DMAINEN and DMAOUTEN bits of the AES_CR register.

Alternative single ECB/CBC decryption using Mode 4
The sequence of events to perform a single round of ECB/CBC decryption using Mode 4 is:
1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select the Mode 4 by setting to 11 the MODE[1:0] bitfield of the AES_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES_CR register to 000 or 001, respectively.
3. Select key length of 128 or 256 bits via KEYSIZE bitfield of the AES_CR register.
4. Write the AES_KEYRx registers with the encryption key. Write the AES_IVRx registers if the CBC mode is selected.
5. Enable the AES peripheral by setting the EN bit of the AES_CR register.
6. Write the AES_DINR register four times to input the cipher text (MSB first).
7. Wait until the CCF flag is set in the AES_SR register.
8. Read the AES_DOUTR register four times to get the plain text (MSB first). Then clear the CCF flag by setting the CCFC bit of the AES_CR register.

Note: When mode 4 is selected mode 3 cannot be used.
In mode 4, the AES_KEYRx registers contain the encryption key during all phases of the processing. No derivation key is stored in these registers. It is stored internally in AES.

19.4.9 AES counter (CTR) mode

Overview
The counter mode (CTR) uses AES as a key-stream generator. The generated keys are then XOR-ed with the plaintext to obtain the ciphertext.

CTR chaining is defined in NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation. A typical message construction in CTR mode is given in Figure 90.
The structure of this message is:

- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter should be set to 1.

- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

### CTR encryption and decryption

*Figure 91* and *Figure 92* describe the CTR encryption and decryption process, respectively, as implemented in the AES peripheral. The CTR mode is selected by writing 010 to the CHMOD[2:0] bitfield of AES_CR register.

---

**Figure 91. CTR encryption**

**Figure 92. CTR decryption**
In CTR mode, the cryptographic core output (also called keystream) Ox is XOR-ed with relevant input block (Px’ for encryption, Cx’ for decryption), to produce the correct output block (Cx’ for encryption, Px’ for decryption). Initialization vectors in AES must be initialized as shown in Table 91.

Table 91. CTR mode initialization vector definition

<table>
<thead>
<tr>
<th>AES_IVR3[31:0]</th>
<th>AES_IVR2[31:0]</th>
<th>AES_IVR1[31:0]</th>
<th>AES_IVR0[31:0]</th>
</tr>
</thead>
</table>

Unlike in CBC mode that uses the AES_IVRx registers only once when processing the first data block, in CTR mode AES_IVRx registers are used for processing each data block, and the AES peripheral increments the counter bits of the initialization vector (leaving the nonce bits unchanged).

CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that is then XOR-ed with the plaintext (CTR encryption) or ciphertext (CTR decryption) input. In CTR mode, the MODE[1:0] bitfield settings 11, 10 or 00 default all to encryption mode, and the setting 01 (key derivation) is forbidden.
The sequence of events to perform an encryption or a decryption in CTR chaining mode:

1. Ensure that AES is disabled (the EN bit of the AES_CR must be 0).
2. Select CTR chaining mode by setting to 010 the CHMOD[2:0] bitfield of the AES_CR register. Set MODE[1:0] bitfield to any value other than 01.
3. Initialize the AES_KEYRx registers, and load the AES_IVRx registers as described in Table 91.
4. Set the EN bit of the AES_CR register, to start encrypting the current counter (EN is automatically reset when the calculation finishes).
5. If it is the last block, pad the data with zeros to have a complete block, if needed.
6. Append data in AES, and read the result. The three possible scenarios are described in Section 19.4.4: AES procedure to perform a cipher operation.
7. Repeat the previous step till the second-last block is processed. For the last block, apply the two previous steps and discard the bits that are not part of the payload (if the size of the significant data in the last input block is less than 16 bytes).

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher priority message, and resume the message that was interrupted. Detailed CBC suspend/resume sequence is described in Section 19.4.8: AES basic chaining modes (ECB, CBC).

Note: Like for CBC mode, the AES_IVRx registers must be reloaded during the resume operation.

19.4.10 AES Galois/counter mode (GCM)

Overview

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, GCM algorithm is based on AES counter mode. It uses a multiplier over a fixed finite field to generate the tag.

GCM chaining is defined in NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC. A typical message construction in GCM mode is given in Figure 93.

![Figure 93. Message construction in GCM](image-url)
The message has the following structure:

- **16-byte initial counter block (ICB),** composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. Note that the GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.

- **Authenticated header AAD** (also known as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} - 1$ bits. This part of the message is only authenticated, not encrypted.

- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed $2^{32} - 2$ 128-bit blocks.

- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in **Table 92.**

The GCM standard specifies that ciphertext C has the same bit length as the plaintext P. When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Input data</td>
<td>0x0</td>
<td>AAD length[31:0]</td>
<td>0x0</td>
<td>Payload length[31:0]</td>
</tr>
</tbody>
</table>
GCM processing

Figure 94 describes the GCM implementation in the AES peripheral. The GCM is selected by writing 011 to the CHMOD[2:0] bitfield of the AES_CR register.

Figure 94. GCM authenticated encryption

The mechanism for the confidentiality of the plaintext in GCM mode is similar to that in the Counter mode, with a particular increment function (denoted 32-bit increment) that generates the sequence of input counter blocks.

AES_IVRx registers keeping the counter block of data are used for processing each data block. The AES peripheral automatically increments the Counter[31:0] bitfield. The first counter block (CB1) is derived from the initial counter block ICB by the application software (see Table 93).

Table 93. GCM mode IVI bitfield initialization

<table>
<thead>
<tr>
<th>Register</th>
<th>AES_IVR3[31:0]</th>
<th>AES_IVR2[31:0]</th>
<th>AES_IVR1[31:0]</th>
<th>AES_IVR0[31:0]</th>
</tr>
</thead>
</table>

Note: In GCM mode, the settings 01 and 11 of the MODE[1:0] bitfield are forbidden.
The authentication mechanism in GCM mode is based on a hash function called \( \text{GF2mul} \) that performs multiplication by a fixed parameter, called hash subkey (H), within a binary Galois field.

A GCM message is processed through the following phases, further described in next subsections:

- **Init phase**: AES prepares the GCM hash subkey (H).
- **Header phase**: AES processes the additional authenticated data (AAD), with hash computation only.
- **Payload phase**: AES processes the plaintext (P) with hash computation, counter block encryption and data XOR-ing. It operates in a similar way for ciphertext (C).
- **Final phase**: AES generates the authenticated tag (T) using the last block of the message.

**GCM init phase**

During this first step, the GCM hash subkey (H) is calculated and saved internally, to be used for processing all the blocks. The recommended sequence is:

1. Ensure that AES is disabled (the EN bit of the AES_CR must be 0).
2. Select GCM chaining mode, by setting to 011 the CHMOD[2:0] bitfield of the AES_CR register, and optionally, set the DATATYPE[1:0] bitfield.
3. Indicate the Init phase, by setting to 00 the GCMPH[1:0] bitfield of the AES_CR register.
4. Set the MODE[1:0] bitfield of the AES_CR register to 00 or 10. Although the bitfield is only used in payload phase, it is recommended to set it in the Init phase and keep it unchanged in all subsequent phases.
5. Initialize the AES_KEYRx registers with a key, and initialize AES_IVRx registers with the information as defined in Table 93.
6. Start the calculation of the hash key, by setting to 1 the EN bit of the AES_CR register (EN is automatically reset when the calculation finishes).
7. Wait until the end of computation, indicated by the CCF flag of the AES_SR transiting to 1. Alternatively, use the corresponding interrupt.
8. Clear the CCF flag of the AES_SR register, by setting to 1 the CCFC bit of the AES_CR register.

**GCM header phase**

This phase coming after the GCM Init phase must be completed before the payload phase. The sequence to execute, identical for encryption and decryption, is:

1. Indicate the header phase, by setting to 01 the GCMPH[1:0] bitfield of the AES_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. Enable the AES peripheral by setting the EN bit of the AES_CR register.
3. If it is the last block and the AAD size in the block is inferior to 128 bits, pad the remainder of the block with zeros. Then append the data block into AES in one of ways described in Section 19.4.4: AES procedure to perform a cipher operation on page 436.
4. Repeat the step 3 until the last additional authenticated data block is processed.

**Note:** The header phase can be skipped if there is no AAD, that is, \( \text{Len}(\text{A}) = 0 \).
GCM payload phase
This phase, identical for encryption and decryption, is executed after the GCM header phase. During this phase, the encrypted/decrypted payload is stored in the AES_DOUTR register. The sequence to execute is:
1. Indicate the payload phase, by setting to 10 the GCMPH[1:0] bitfield of the AES_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. If the header phase was skipped, enable the AES peripheral by setting the EN bit of the AES_CR register.
3. If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is inferior to 128 bits, pad the remainder of the block with zeros.
4. Append the data block into AES in one of ways described in Section 19.4.4: AES procedure to perform a cipher operation on page 436, and read the result.
5. Repeat the previous step till the second-last plaintext block is encrypted or till the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), execute the two previous steps. For the last block, discard the bits that are not part of the payload when the last block size is less than 16 bytes.

Note: The payload phase can be skipped if there is no payload data, that is, Len(C) = 0 (see GMAC mode).

GCM final phase
In this last phase, the AES peripheral generates the GCM authentication tag and stores it in the AES_DOUTR register. The sequence to execute is:
1. Indicate the final phase, by setting to 11 the GCMPH[1:0] bitfield of the AES_CR register.
2. Compose the data of the block, by concatenating the AAD bit length and the payload bit length, as shown in Table 92. Write the block into the AES_DINR register.
3. Wait until the end of computation, indicated by the CCF flag of the AES_SR transiting to 1.
4. Get the GCM authentication tag, by reading the AES_DOUTR register four times.
5. Clear the CCF flag in the AES_SR register, by setting to 1 the CCFC bit of the AES_CR register.
6. Disable the AES peripheral, by clearing the bit EN of the AES_CR register. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data must be inserted normally (no swapping).
When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode
To suspend the processing of a message, proceed as follows:
1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES_CR register. If DMA is not used, make sure that the current computation is completed, which is indicated by the CCF flag of the AES_SR register set to 1.
2. In the payload phase, if DMA is not used, read four times the AES_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag is set in the
AES_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES_CR register.

3. Clear the CCF flag of the AES_SR register, by setting to 1 the CCFC bit of the AES_CR register. In the payload phase (encryption mode only), verify that the BUSY flag of the AES_SR register is cleared, to ensure that GF2mul hash function is completed.

4. Save the AES_SUSPxR registers in the memory, where x is from 0 to 7.

5. In the payload phase, save the AES_IVRx registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.

6. Disable the AES peripheral, by clearing the EN bit of the AES_CR register.

7. Save the current AES configuration in the memory, excluding the initialization vector registers AES_IVRx. Key registers do not need to be saved as the original key value is known by the application.

8. If DMA is used, save the DMA controller status (pointers for IN data transfers, number of remaining bytes, and so on). In the payload phase, pointers for OUT data transfers must also be saved.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the rest of the FIFO IN transfers. In the payload phase, the rest of the FIFO OUT transfers must also be configured in the DMA controller.

2. Ensure that the AES peripheral is disabled (the EN bit of the AES_CR register must be 0).

3. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPxR registers, where x is from 0 to 7.

4. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers. In the header phase, write initial setting values back into the AES_IVRx registers.

5. Restore the initial setting values in the AES_CR and AES_KEYRx registers.

6. Enable the AES peripheral by setting the EN bit of the AES_CR register.

7. If DMA is used, enable AES DMA requests by setting the DMAINEN bit (and DMAOUTEN bit if in payload phase) of the AES_CR register.

### 19.4.11 AES Galois message authentication code (GMAC)

#### Overview

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code). It is based on GCM algorithm, as defined in NIST Special Publication 800-38D, *Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*.
A typical message construction for GMAC is given in Figure 95.

**Figure 95. Message construction in GMAC mode**

![Message construction in GMAC mode](image)

AES GMAC processing

*Figure 96* describes the GMAC mode implementation in the AES peripheral. This mode is selected by writing 011 to the CHMOD[2:0] bitfield of the AES_CR register.

**Figure 96. GMAC authentication mode**

![GMAC authentication mode](image)

The GMAC algorithm corresponds to the GCM algorithm applied on a message only containing a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

**Suspend/resume operations in GMAC**

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.
19.4.12 AES counter with CBC-MAC (CCM)

Overview

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES in counter mode. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve this CBC-MAC as an authentication mode outside the context of the CCM specification.

CCM chaining is specified in NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality. A typical message construction for CCM is given in Figure 97.

Figure 97. Message construction in CCM mode

The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q**: a bit string representation of the octet length of P (Len(P))
  - **Nonce (N)**: a single-use value (that is, a new nonce should be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags**: most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values t (MAC length expressed in bytes) and Q (plaintext length such that Len(P) < 2^8Q bytes). The counter blocks range associated to Q is equal to 2^8Q-4, that is, if the maximum value of Q is 8, the counter blocks used in cipher shall be on 60 bits.

- **16-byte blocks (B)** associated to the Associated Data (A).

This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see Figure 97). The
standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:

- If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
- If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as $0xff || 0xfe || [a]_{32}$, that is, on six bytes.
- If $2^{32} < a < 2^{64}$, then it is encoded as $0xff || 0xff || [a]_{64}$, that is, on ten bytes.

- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see Figure 97).

- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

**Note:** *CCM chaining mode can also be used with associated data only (that is, no payload).*

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

- **N**: 101111213 141516 (Len(N)= 56 bits or 7 bytes)
- **A**: 00010203 04050607 08090A0B (Len(A)= 64 bits or 8 bytes)
- **P**: 20212223 (Len(P)= 32 bits or 4 bytes)
- **T**: 6084341B (Len(T)= 32 bits or t = 4)
- **B0**: 4F101112 13141516 00000000 00000004
- **B1**: 00080001 02030405 06070000 00000000
- **B2**: 20212223 00000000 00000001
- **CTR0**: 0710111213 141516 00000000 00000000
- **CTR1**: 0710111213 141516 00000000 00000000

Generation of formatted input data blocks Bx (especially B0 and B1) must be managed by the application.
CCM processing

Figure 98 describes the CCM implementation within the AES peripheral (encryption example). This mode is selected by writing 100 into the CHMOD[2:0] bitfield of the AES_CR register.

The data input to the generation-encryption process are a valid nonce, a valid payload string, and a valid associated data string, all properly formatted. The CBC chaining mechanism is applied to the formatted plaintext data to generate a MAC, with a known length. Counter mode encryption that requires a sufficiently long sequence of counter blocks as input, is applied to the payload string and separately to the MAC. The resulting ciphertext C is the output of the generation-encryption process on plaintext P.

AES_IVRx registers are used for processing each data block, AES automatically incrementing the CTR counter with a bit length defined by the first block B0. Table 94 shows how the application must load the B0 data.

Note: The AES peripheral in CCM mode supports counters up to 64 bits, as specified by NIST.
Table 94. Initialization of AES_IVRx registers in CCM mode

<table>
<thead>
<tr>
<th>Register</th>
<th>AES_IVR3[31:0]</th>
<th>AES_IVR2[31:0]</th>
<th>AES_IVR1[31:0]</th>
<th>AES_IVR0[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input data</td>
<td>B0[31:0]</td>
<td>B0[63:32]</td>
<td>B0[95:64]</td>
<td>B0[127:96]</td>
</tr>
</tbody>
</table>

Note: In CCM mode the MODE[1:0] bitfield settings 01 and 11 (key derivation) are forbidden.

A CCM message is processed through the following phases, further described in next subsections:
- **Init phase:** AES processes the first block and prepares the first counter block.
- **Header phase:** AES processes associated data (A), with tag computation only.
- **Payload phase:** IP processes plaintext (P), with tag computation, counter block encryption, and data XOR-ing. It works in a similar way for ciphertext (C).
- **Final phase:** AES generates the message authentication code (MAC).

**CCM Init phase**

In this phase, the first block B0 of the CCM message is written into the AES_IVRx register. The AES_DOUTR register does not contain any output data. The recommended sequence is:
1. Ensure that the AES peripheral is disabled (the EN bit of the AES_CR must be 0).
2. Select CCM chaining mode, by setting to 100 the CHMOD[2:0] bitfield of the AES_CR register, and optionally, set the DATATYPE[1:0] bitfield.
3. Indicate the Init phase, by setting to 00 the GCMPH[1:0] bitfield of the AES_CR register.
4. Set the MODE[1:0] bitfield of the AES_CR register to 00 or 10. Although the bitfield is only used in payload phase, it is recommended to set it in the Init phase and keep it unchanged in all subsequent phases.
5. Initialize the AES_KEYRx registers with a key, and initialize AES_IVRx registers with B0 data as described in Table 94.
6. Start the calculation of the counter, by setting to 1 the EN bit of the AES_CR register (EN is automatically reset when the calculation finishes).
7. Wait until the end of computation, indicated by the CCF flag of the AES_SR transiting to 1. Alternatively, use the corresponding interrupt.
8. Clear the CCF flag in the AES_SR register, by setting to 1 the CCFC bit of the AES_CR register.

**CCM header phase**

This phase coming after the GCM Init phase must be completed before the payload phase. During this phase, the AES_DOUTR register does not contain any output data.

The sequence to execute, identical for encryption and decryption, is:
1. Indicate the header phase, by setting to 01 the GCMPH[1:0] bitfield of the AES_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. If the header phase was skipped, enable the AES peripheral by setting the EN bit of the AES_CR register.
3. If it is the last block and the AAD size in the block is inferior to 128 bits, pad the remainder of the block with zeros. Then append the data block into AES in one of ways.
described in Section 19.4.4: AES procedure to perform a cipher operation on page 436.

4. Repeat the step 3 until the last additional authenticated data block is processed.

**Note:** The header phase can be skipped if there is no associated data, that is, Len(A) = 0. The first block of the associated data (B1) must be formatted by software, with the associated data length.

**CCM payload phase (encryption or decryption)**

This phase, identical for encryption and decryption, is executed after the CCM header phase. During this phase, the encrypted/decrypted payload is stored in the AES_DOUTR register. The sequence to execute is:

1. Indicate the payload phase, by setting to 10 the GCMPH[1:0] bitfield of the AES_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. If the header phase was skipped, enable the AES peripheral by setting the EN bit of the AES_CR register.
3. If it is the last data block to encrypt and the plaintext size in the block is inferior to 128 bits, pad the remainder of the block with zeros.
4. Append the data block into AES in one of ways described in Section 19.4.4: AES procedure to perform a cipher operation on page 436, and read the result.
5. Repeat the previous step till the second-last plaintext block is encrypted or till the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), apply the two previous steps. For the last block, discard the data that is not part of the payload when the last block size is less than 16 bytes.

**Note:** The payload phase can be skipped if there is no payload data, that is, Len(P) = 0 or Len(P) = Len(T).

Remove LSB_{Len(T)}(C) encrypted tag information when decrypting ciphertext C.

**CCM final phase**

In this last phase, the AES peripheral generates the GCM authentication tag and stores it in the AES_DOUTR register. The sequence to execute is:

1. Indicate the final phase, by setting to 11 the GCMPH[1:0] bitfield of the AES_CR register.
2. Wait until the end-of-computation flag CCF of the AES_SR register is set.
3. Read four times the AES_DOUTR register: the output corresponds to the CCM authentication tag.
4. Clear the CCF flag of the AES_SR register by setting the CCFC bit of the AES_CR register.
5. Disable the AES peripheral, by clearing the EN bit of the AES_CR register.
6. For authenticated decryption, compare the generated encrypted tag with the encrypted tag padded in the ciphertext.

**Note:** In this final phase, data is read normally (no swapping).

When transiting from the header phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Application must mask the authentication tag output with tag length to obtain a valid tag.
Suspend/resume operations in CCM mode

To suspend the processing of a message in header or payload phase, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES_CR register. If DMA is not used, make sure that the current computation is completed, which is indicated by the CCF flag of the AES_SR register set to 1.

2. In the payload phase, if DMA is not used, read four times the AES_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag is set in the AES_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES_CR register.

3. Clear the CCF flag of the AES_SR register, by setting to 1 the CCFC bit of the AES_CR register.

4. Save the AES_SUSPxR registers (where x is from 0 to 7) in the memory.

5. Save the AES_IVRx registers as, during the data processing, they changed from their initial values.

6. Disable the AES peripheral, by clearing the EN bit of the AES_CR register.

7. Save the current AES configuration in the memory, excluding the initialization vector registers AES_IVRx. Key registers do not need to be saved as the original key value is known by the application.

8. If DMA is used, save the DMA controller status (pointers for IN data transfers, number of remaining bytes, and so on). In the payload phase, pointers for OUT data transfers must also be saved.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the rest of the FIFO IN transfers. In the payload phase, the rest of the FIFO OUT transfers must also be configured in the DMA controller.

2. Ensure that the AES peripheral is disabled (the EN bit of the AES_CR register must be 0).

3. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPxR registers (where x is from 0 to 7).

4. Write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers.

5. Restore the initial setting values in the AES_CR and AES_KEYRx registers.

6. Enable the AES peripheral by setting the EN bit of the AES_CR register.

7. If DMA is used, enable AES DMA requests by setting to 1 the DMAINEN bit (and DMAOUTEN bit if in payload phase) of the AES_CR register.
19.4.13 AES data registers and data swapping

Data input and output

A 128-bit data block is entered into the AES peripheral with four successive 32-bit word writes into the AES_DINR register (bitfield DIN[127:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 128-bit data block is retrieved from the AES peripheral with four successive 32-bit word reads from the AES_DOUTR register (bitfield DOUT[127:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The 32-bit data word for AES_DINR register or from AES_DOUTR register is organized in big endian order, that is:

- the most significant byte of a word to write into AES_DINR must be put on the lowest address out of the four adjacent memory locations keeping the word to write, or
- the most significant byte of a word read from AES_DOUTR goes to the lowest address out of the four adjacent memory locations receiving the word

For using DMA for input data block write into AES, the four words of the input block must be stored in the memory consecutively and in big-endian order, that is, the most significant word on the lowest address. See Section 19.4.16: AES DMA interface.

Data swapping

The AES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the AES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the AES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through the DATATYPE[1:0] bitfield of the AES_CR register. The selection applies both to the input and the output of the AES core.

For different data swap types, Figure 99 shows the construction of AES processing core input buffer data P127..0, from the input data entered through the AES_DINR register, or the construction of the output data available through the AES_DOUTR register, from the AES processing core output buffer data P127..0.
Note: The data in AES key registers (AES_KEYRx) and initialization registers (AES_IVR Rx) are not sensitive to the swap mode selection.

Data padding

Figure 99 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 48 message bits, with DATATYPE[1:0] = 01
- 56 message bits, with DATATYPE[1:0] = 10
- 34 message bits, with DATATYPE[1:0] = 11
19.4.14 AES key registers

The AES_KEYRx registers store the encryption or decryption key bitfield KEY[127:0] or KEY[255:0]. The data to write to or to read from each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address.

The key is spread over the eight registers as shown in Table 95.

Table 95. Key endianness in AES_KEYRx registers (128- or 256-bit key length)

<table>
<thead>
<tr>
<th>AES_KEYR7 [31:0]</th>
<th>AES_KEYR6 [31:0]</th>
<th>AES_KEYR5 [31:0]</th>
<th>AES_KEYR4 [31:0]</th>
<th>AES_KEYR3 [31:0]</th>
<th>AES_KEYR2 [31:0]</th>
<th>AES_KEYR1 [31:0]</th>
<th>AES_KEYR0 [31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>KEY[127:96]</td>
<td>KEY[95:64]</td>
<td>KEY[63:32]</td>
<td>KEY[31:0]</td>
</tr>
</tbody>
</table>

The key for encryption or decryption may be written into these registers when the AES peripheral is disabled.

The key registers are not affected by the data swapping controlled by DATATYPE[1:0] bitfield of the AES_CR register.

19.4.15 AES initialization vector registers

The four AES_IVRx registers keep the initialization vector input bitfield IVI[127:0]. The data to write to or to read from each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address. The registers are also ordered from lowest address (AES_IVR0) to highest address (AES_IVR3).

The signification of data in the bitfield depends on the chaining mode selected. When used, the bitfield is updated upon each computation cycle of the AES core.

Write operations to the AES_IVRx registers when the AES peripheral is enabled have no effect to the register contents. For modifying the contents of the AES_IVRx registers, the EN bit of the AES_CR register must first be cleared.

Reading the AES_IVRx registers returns the latest counter value (useful for managing suspend mode).

The AES_IVRx registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield of the AES_CR register.

19.4.16 AES DMA interface

The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the AES_CR register.

Data input using DMA

Setting the DMAINEN bit of the AES_CR register enables DMA writing into AES. The AES peripheral then initiates a DMA request during the input phase each time it requires to write a 128-bit block (quadruple word) to the AES_DINR register, as shown in Figure 100.
Table 96. DMA channel configuration for memory-to-AES data transfer

<table>
<thead>
<tr>
<th>DMA channel control register field</th>
<th>Recommended configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer size</td>
<td>Message length: a multiple of 128 bits. According to the algorithm and the mode selected, special padding/ciphertext stealing might be required. For example, in case of AES GCM encryption or AES CCM decryption, a DMA transfer must not include the last block. Refer to Section 19.4.4: AES procedure to perform a cipher operation for details.</td>
</tr>
<tr>
<td>Source burst size (memory)</td>
<td>Single</td>
</tr>
<tr>
<td>Destination burst size (peripheral)</td>
<td>Single</td>
</tr>
<tr>
<td>DMA FIFO size</td>
<td>AES FIFO_size = 4 bytes.</td>
</tr>
<tr>
<td>Source transfer width (memory)</td>
<td>32-bit words</td>
</tr>
<tr>
<td>Destination transfer width (peripheral)</td>
<td>32-bit words</td>
</tr>
<tr>
<td>Source address increment (memory)</td>
<td>Yes, after each 32-bit transfer</td>
</tr>
<tr>
<td>Destination address increment (peripheral)</td>
<td>Fixed address of AES_DINR (no increment)</td>
</tr>
</tbody>
</table>

**Note:** According to the algorithm and the mode selected, special padding/ciphertext stealing might be required. For example, in case of AES GCM encryption or AES CCM decryption, a DMA transfer must not include the last block. Refer to Section 19.4.4: AES procedure to perform a cipher operation for details.

Figure 100. DMA transfer of a 128-bit data block during input phase
Data output using DMA

Setting the DMAOUTEN bit of the AES_CR register enables DMA reading from AES. The AES peripheral then initiates a DMA request during the Output phase each time it requires to read a 128-bit block (quadruple word) to the AES_DINR register, as shown in Figure 101.

Table 97. DMA channel configuration for AES-to-memory data transfer

<table>
<thead>
<tr>
<th>DMA channel control register field</th>
<th>Recommended configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer size</td>
<td>It is the message length multiple of AES block size (4 words). According to the case extra bytes will have to be discarded.</td>
</tr>
<tr>
<td>Source burst size (peripheral)</td>
<td>Single</td>
</tr>
<tr>
<td>Destination burst size (memory)</td>
<td>Single</td>
</tr>
<tr>
<td>DMA FIFIO size</td>
<td>AES FIFO_size = 4 bytes</td>
</tr>
<tr>
<td>Source transfer width (peripheral)</td>
<td>32-bit words</td>
</tr>
<tr>
<td>Destination transfer width (memory)</td>
<td>32-bit words</td>
</tr>
<tr>
<td>Source address increment (peripheral)</td>
<td>Fixed address of AES_DINR (no increment)</td>
</tr>
<tr>
<td>Destination address increment (memory)</td>
<td>Yes, after each 32-bit transfer</td>
</tr>
</tbody>
</table>

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Figure 101. DMA transfer of a 128-bit data block during output phase
DMA operation in different operating modes

DMA operations are usable when Mode 1 (encryption) or Mode 3 (decryption) are selected via the MODE[1:0] bitfield of the register AES_CR. As in Mode 2 (key derivation) the AES_KEYRx registers must be written by software, enabling the DMA transfer through the DMAINEN and DMAOUTEN bits of the AES_CR register have no effect in that mode.

DMA single requests are generated by AES until it is disabled. So, after the data output phase at the end of processing of a 128-bit data block, AES switches automatically to a new data input phase for the next data block, if any.

When the data transferring between AES and memory is managed by DMA, the CCF flag is not relevant and can be ignored (left set) by software. It must only be cleared when transiting back to data transferring managed by software. See Suspend/resume operations in ECB/CBC modes in Section 19.4.8: AES basic chaining modes (ECB, CBC) as example.

19.4.17 AES error management

The read error flag (RDERR) and write error flag (WRERR) of the AES_SR register are set when an unexpected read or write operation, respectively, is detected. An interrupt can be generated if the error interrupt enable (ERRIE) bit of the AES_CR register is set. For more details, refer to Section 19.5: AES interrupts.

Note: AES is not disabled after an error detection and continues processing.

AES can be re-initialized at any moment by clearing then setting the EN bit of the AES_CR register.

Read error flag (RDERR)

When an unexpected read operation is detected during the computation phase or during the input phase, the AES read error flag (RDERR) is set in the AES_SR register. An interrupt is generated if the ERRIE bit of the AES_CR register is set.

The RDERR flag is cleared by setting the corresponding ERRC bit of the AES_CR register.

Write error flag (WDERR)

When an unexpected write operation is detected during the computation phase or during the output phase, the AES write error flag (WRERR) is set in the AES_SR register. An interrupt is generated if the ERRIE bit of the AES_CR register is set.

The WDERR flag is cleared by setting the corresponding ERRC bit of the AES_CR register.

19.5 AES interrupts

There are three individual maskable interrupt sources generated by the AES peripheral, to signal the following events:

- computation completed
- read error, see Section 19.4.17
- write error, see Section 19.4.17

These three sources are combined into a common interrupt signal aes_it that connects to NVIC (nested vectored interrupt controller).
Each AES interrupt source can individually be enabled/disabled, by setting/clearing the corresponding enable bit of the AES_CR register. See Figure 102.

The status of the individual maskable interrupt sources is read from the AES_SR register. Table 98 gives a summary of the interrupt sources, their event flags and enable bits.

### Table 98. AES interrupt requests

<table>
<thead>
<tr>
<th>AES interrupt event</th>
<th>Event flag</th>
<th>Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>computation completed flag</td>
<td>CCF</td>
<td>CCFIE</td>
</tr>
<tr>
<td>read error flag</td>
<td>RDERR</td>
<td>ERRIE</td>
</tr>
<tr>
<td>write error flag</td>
<td>WRERR</td>
<td>ERRIE</td>
</tr>
</tbody>
</table>

#### 19.6 AES processing latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

### Table 99. Processing latency (in clock cycle) for ECB, CBC and CTR

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Algorithm</th>
<th>Input phase + FSM set</th>
<th>Computation phase</th>
<th>Output phase</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>128-bit</td>
<td>Mode 1: Encryption</td>
<td>ECB, CBC, CTR</td>
<td>9</td>
<td>38</td>
<td>4</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>Mode 2: Key derivation</td>
<td>-</td>
<td>-</td>
<td>59</td>
<td>-</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>Mode 3: Decryption</td>
<td>ECB, CBC, CTR</td>
<td>9</td>
<td>38</td>
<td>4</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>Mode 4: Key derivation then decryption</td>
<td>ECB, CBC</td>
<td>9</td>
<td>93</td>
<td>4</td>
<td>106</td>
</tr>
<tr>
<td>256-bit</td>
<td>Mode 1: Encryption</td>
<td>ECB, CBC, CTR</td>
<td>13</td>
<td>58</td>
<td>4</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>Mode 2: Key derivation</td>
<td>-</td>
<td>-</td>
<td>82</td>
<td>-</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>Mode 3: Decryption</td>
<td>ECB, CBC, CTR</td>
<td>13</td>
<td>58</td>
<td>4</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>Mode 4: Key derivation then decryption</td>
<td>ECB, CBC</td>
<td>13</td>
<td>128</td>
<td>4</td>
<td>145</td>
</tr>
</tbody>
</table>
Table 100. Processing latency for GCM and CCM (in clock cycle)

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Algorithm</th>
<th>Init Phase</th>
<th>Header phase</th>
<th>Payload phase</th>
<th>Tag phase</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mode 1: Encryption/Mode 3: Decryption</td>
<td>GCM</td>
<td>64</td>
<td>35</td>
<td>51</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCM</td>
<td>63</td>
<td>55</td>
<td>114</td>
<td>58</td>
</tr>
<tr>
<td>128-bit</td>
<td>Mode 1: Encryption/Mode 3: Decryption</td>
<td>GCM</td>
<td>88</td>
<td>35</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCM</td>
<td>87</td>
<td>79</td>
<td>162</td>
<td>82</td>
</tr>
<tr>
<td>256-bit</td>
<td>Mode 1: Encryption/Mode 3: Decryption</td>
<td>GCM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle). This applies to all header/payload/tag phases.
19.7 AES registers

19.7.1 AES control register (AES_CR)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 23:20</td>
<td>NPBLB[3:0]: Number of padding bytes in last block</td>
</tr>
<tr>
<td></td>
<td>The bitfield sets the number of padding bytes in last block of payload:</td>
</tr>
<tr>
<td></td>
<td>0000: All bytes are valid (no padding)</td>
</tr>
<tr>
<td></td>
<td>0001: Padding for one least-significant byte of last block</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>1111: Padding for 15 least-significant bytes of last block</td>
</tr>
<tr>
<td>Bit 19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 18</td>
<td>KEYSIZE: Key size selection</td>
</tr>
<tr>
<td></td>
<td>This bitfield defines the length of the key used in the AES cryptographic core, in bits:</td>
</tr>
<tr>
<td></td>
<td>0: 128</td>
</tr>
<tr>
<td></td>
<td>1: 256</td>
</tr>
<tr>
<td></td>
<td>The bit value change is allowed only when AES is disabled, so as to avoid an unpredictable behavior.</td>
</tr>
<tr>
<td>Bit 17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 16</td>
<td>CHMOD[2]: Chaining mode selection, bit [2]</td>
</tr>
<tr>
<td></td>
<td>Refer to the bits [5:6] of the register for the description of the CHMOD[2:0] bitfield</td>
</tr>
<tr>
<td>Bit 15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 14:13</td>
<td>GCMPH[1:0]: GCM or CCM phase selection</td>
</tr>
<tr>
<td></td>
<td>This bitfield selects the phase of GCM, GMAC or CCM algorithm:</td>
</tr>
<tr>
<td></td>
<td>00: Init phase</td>
</tr>
<tr>
<td></td>
<td>01: Header phase</td>
</tr>
<tr>
<td></td>
<td>10: Payload phase</td>
</tr>
<tr>
<td></td>
<td>11: Final phase</td>
</tr>
<tr>
<td></td>
<td>The bitfield has no effect if other than GCM, GMAC or CCM algorithms are selected (through the ALGOMODE bitfield).</td>
</tr>
</tbody>
</table>
Bit 12 DMAOUTEN: DMA output enable
This bit enables/disables data transferring with DMA, in the output phase:
0: Disable
1: Enable

When the bit is set, DMA requests are automatically generated by AES during the output data phase. This feature is only effective when Mode 1 or Mode 3 is selected through the MODE[1:0] bitfield. It is not effective for Mode 2 (key derivation).
Usage of DMA with Mode 4 (single decryption) is not recommended.

Bit 11 DMAINEN: DMA input enable
This bit enables/disables data transferring with DMA, in the input phase:
0: Disable
1: Enable

When the bit is set, DMA requests are automatically generated by AES during the input data phase. This feature is only effective when Mode 1 or Mode 3 is selected through the MODE[1:0] bitfield. It is not effective for Mode 2 (key derivation).
Usage of DMA with Mode 4 (single decryption) is not recommended.

Bit 10 ERRIE: Error interrupt enable
This bit enables or disables (masks) the AES interrupt generation when RDERR and/or WRERR is set:
0: Disable (mask)
1: Enable

Bit 9 CCFIE: CCF interrupt enable
This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set:
0: Disable (mask)
1: Enable

Bit 8 ERRC: Error flag clear
Upon written to 1, this bit clears the RDERR and WRERR error flags in the AES_SR register:
0: No effect
1: Clear RDERR and WRERR flags

Reading the flag always returns zero.

Bit 7 CCFC: Computation complete flag clear
Upon written to 1, this bit clears the computation complete flag (CCF) in the AES_SR register:
0: No effect
1: Clear CCF

Reading the flag always returns zero.
Bits 6:5  **CHMOD[1:0]**: Chaining mode selection, bits [1:0]
   These bits, together with the bit CHMOD[2] (see bit 16 of this register), form CHMOD[2:0] bitfield that selects the AES chaining mode:
   - 000: Electronic codebook (ECB)
   - 001: Cipher-Block Chaining (CBC)
   - 010: Counter Mode (CTR)
   - 011: Galois Counter Mode (GCM) and Galois Message Authentication Code (GMAC)
   - 100: Counter with CBC-MAC (CCM)
   >100: Reserved

   The bitfield value change is allowed only when AES is disabled, so as to avoid an unpredictable behavior.

Bits 4:3  **MODE[1:0]**: AES operating mode
   This bitfield selects the AES operating mode:
   - 00: Mode 1: encryption
   - 01: Mode 2: key derivation (or key preparation for ECB/CBC decryption)
   - 10: Mode 3: decryption
   - 11: Mode 4: key derivation then single decryption

   The bitfield value change is allowed only when AES is disabled, so as to avoid an unpredictable behavior. Any attempt to selecting Mode 4 while either ECB or CBC chaining mode is not selected, defaults to effective selection of Mode 3. It is not possible to select a Mode 3 following a Mode 4.

Bits 2:1  **DATATYPE[1:0]**: Data type selection
   This bitfield defines the format of data written in the AES_DINR register or read from the AES_DOUTR register, through selecting the mode of data swapping:
   - 00: None
   - 01: Half-word (16-bit)
   - 10: Byte (8-bit)
   - 11: Bit

   For more details, refer to Section 19.4.13: AES data registers and data swapping.

   The bitfield value change is allowed only when AES is disabled, so as to avoid an unpredictable behavior.

Bit 0  **EN**: AES enable
   This bit enables/disables the AES peripheral:
   - 0: Disable
   - 1: Enable

   At any moment, clearing then setting the bit re-initializes the AES peripheral.

   This bit is automatically cleared by hardware when the key preparation process ends (Mode 2).

19.7.2  **AES status register (AES_SR)**
   Address offset: 0x04
   Reset value: 0x0000 0000
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

**Bit 3 BUSY:** Busy

This flag indicates whether AES is idle or busy during GCM payload encryption phase:

- 0: Idle
- 1: Busy

The flag is controlled by hardware. When the flag indicates “idle”, the current message processing may be suspended to process a higher-priority message.

This flag is effective only in GCM payload encryption phase. In other chaining modes, or in GCM phases other than payload encryption, the flag must be ignored.

**Bit 2 WRERR:** Write error

This flag indicates the detection of an unexpected write operation to the AES_DINR register (during computation or data output phase):

- 0: Not detected
- 1: Detected

The flag is set by hardware. It is cleared by software upon setting the ERRC bit of the AES_CR register.

Upon the flag setting, an interrupt is generated if enabled through the ERRIE bit of the AES_CR register.

The flag setting has no impact on the AES operation.

The flag is not effective when key derivation mode, or GCM/CCM init phase is selected.

**Bit 1 RDERR:** Read error flag

This flag indicates the detection of an unexpected read operation from the AES_DOUTR register (during computation or data input phase):

- 0: Not detected
- 1: Detected

The flag is set by hardware. It is cleared by software upon setting the ERRC bit of the AES_CR register.

Upon the flag setting, an interrupt is generated if enabled through the ERRIE bit of the AES_CR register.

The flag setting has no impact on the AES operation.

The flag is not effective when key derivation mode, nor GCM/CCM init/header phase is selected.
Bit 0  **CCF**: Computation completed flag
This flag indicates whether the computation is completed:
0: Not completed
1: Completed

The flag is set by hardware upon the completion of the computation. It is cleared by software, upon setting the CCFC bit of the AES_CR register.
Upon the flag setting, an interrupt is generated if enabled through the CCFIE bit of the AES_CR register.
The flag is significant only when the DMAOUTEN bit is 0. It may stay high when DMA_EN is 1.

### 19.7.3  AES data input register (AES_DINR)

Address offset: 0x08
Reset value: 0x0000 0000
Only 32-bit access type is supported.

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

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

Bits 31:0  **DIN[x+31:x]**: One of four 32-bit words of a 128-bit input data block being written into the peripheral
This bitfield feeds a 32-bit input buffer. A 4-fold sequential write to this bitfield during the input phase virtually writes a complete 128-bit block of input data to the AES peripheral. Upon each write, the data from the input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 128-bit input buffer.
The substitution for “x”, from the first to the fourth write operation, is: 96, 64, 32, and 0. In other words, data from the first to the fourth write operation are: DIN[127:96], DIN[95:64], DIN[63:32], and DIN[31:0].
The data signification of the input data block depends on the AES operating mode:
- **Mode 1** (encryption): plaintext
- **Mode 2** (key derivation): the bitfield is not used (AES_KEYRx registers used for input)
- **Mode 3** (decryption) and **Mode 4** (key derivation then single decryption): ciphertext
The data swap operation is described in Section 19.4.13: AES data registers and data swapping on page 463.
19.7.4 AES data output register (AES_DOUTR)

Address offset: 0x0C
Reset value: 0x0000 0000

Only 32-bit access type is supported.

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

Bits 31:0 DOUT[x+31:x]: One of four 32-bit words of a 128-bit output data block being read from the peripheral. This read-only bitfield fetches a 32-bit output buffer. A 4-fold sequential read of this bitfield, upon the computation completion (CCF set), virtually reads a complete 128-bit block of output data from the AES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield. The substitution for DOUT[x+31:x], from the first to the fourth read operation, is: 96, 64, 32, and 0. In other words, data from the first to the fourth read operation are: DOUT[127:96], DOUT[95:64], DOUT[63:32], and DOUT[31:0].

The data signification of the output data block depends on the AES operating mode:
- **Mode 1** (encryption): ciphertext
- **Mode 2** (key derivation): the bitfield is not used (AES_KEYRx registers used for output)
- **Mode 3** (decryption) and **Mode 4** (key derivation then single decryption): plaintext

The data swap operation is described in Section 19.4.13: AES data registers and data swapping on page 463.

19.7.5 AES key register 0 (AES_KEYR0)

Address offset: 0x10
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]  
This bitfield contains the bits [31:0] of the AES encryption or decryption key, depending on the operating mode:
- In **Mode 1** (encryption), **Mode 2** (key derivation) and **Mode 4** (key derivation then single decryption): the value to write into the bitfield is the encryption key.
- In **Mode 3** (decryption): the value to write into the bitfield is the encryption key to be derived before being used for decryption. After writing the encryption key into the bitfield, its reading before enabling AES returns the same value. Its reading after enabling AES and after the CCF flag is set returns the decryption key derived from the encryption key.

**Note:** In mode 4 (key derivation then single decryption) the bitfield always contains the encryption key.

The AES_KEYRx registers may be written only when the AES peripheral is disabled. Refer to Section 19.4.14: AES key registers on page 465 for more details.

### 19.7.6 AES key register 1 (AES_KEYR1)

Address offset: 0x14  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>KEY[63:48]</th>
<th>Bits 31:0</th>
<th>KEY[47:32]</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]  
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

### 19.7.7 AES key register 2 (AES_KEYR2)

Address offset: 0x18  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>KEY[95:80]</th>
<th>Bits 31:0</th>
<th>KEY[79:64]</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]  
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

### 19.7.8 AES key register 3 (AES_KEYR3)

Address offset: 0x1C  
Reset value: 0x0000 0000
19.7.9 AES initialization vector register 0 (AES_IVR0)

Address offset: 0x20
Reset value: 0x0000 0000

Bits 31:0 KE[127:96]: Cryptographic key, bits [127:96]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

19.7.10 AES initialization vector register 1 (AES_IVR1)

Address offset: 0x24
Reset value: 0x0000 0000

Bits 31:0 IVI[31:0]: Initialization vector input, bits [31:0]
Refer to Section 19.4.15: AES initialization vector registers on page 465 for description of the IVI[128:0] bitfield.
The initialization vector is only used in chaining modes other than ECB.
The initialization vector may be written only when the AES peripheral is disabled.

19.7.11 AES initialization vector register 2 (AES_IVR2)

Address offset: 0x28
Reset value: 0x0000 0000

### 19.7.12 AES initialization vector register 3 (AES_IVR3)

Address offset: 0x2C

Reset value: 0x0000 0000

Bits 31:0 IVI[95:64]: Initialization vector input, bits [95:64]
Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

### 19.7.13 AES key register 4 (AES_KEYR4)

Address offset: 0x30

Reset value: 0x0000 0000

Bits 31:0 KEY[159:128]: Cryptographic key, bits [159:128]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

### 19.7.14 AES key register 5 (AES_KEYR5)

Address offset: 0x34

Reset value: 0x0000 0000
19.7.15 AES key register 6 (AES_KEYR6)

Address offset: 0x38
Reset value: 0x0000 0000

Bits 31:0 KEY[191:160]: Cryptographic key, bits 191:160
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

19.7.16 AES key register 7 (AES_KEYR7)

Address offset: 0x3C
Reset value: 0x0000 0000

Bits 31:0 KEY[223:192]: Cryptographic key, bits 223:192
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

19.7.17 AES suspend registers (AES_SUSPxR)

Address offset: 0x040 + x * 0x4, (x = 0 to 7)
Reset value: 0x0000 0000
These registers contain the complete internal register states of the AES processor when the AES processing of the current task is suspended to process a higher-priority task. Upon suspend, the software reads and saves the AES_SUSPxR register contents (where x is from 0 to 7) into memory, before using the AES processor for the higher-priority task. Upon completion, the software restores the saved contents back into the corresponding suspend registers, before resuming the original task.

**Note:** These registers are used only when GCM, GMAC, or CCM chaining mode is selected. These registers can be read only when AES is enabled. Reading these registers while AES is disabled returns 0x0000 0000.

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

Bits 31:0 **SUSPx:** AES suspend

Upon suspend operation, this bitfield of every AES_SUSPxR register takes the value of one of internal AES registers.
# AES register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset Register</th>
<th>Offset Register</th>
<th>Offset Register</th>
<th>Offset Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>AES_CR</td>
<td>AES_SR</td>
<td>AES_DINR</td>
<td>AES_KEYR0</td>
<td>AES_KEYR1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0008</td>
<td>AES_SR</td>
<td></td>
<td>AES_DINR</td>
<td>AES_KEYR0</td>
<td>AES_KEYR1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x000C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0010</td>
<td>AES_KEYR0</td>
<td></td>
<td></td>
<td>AES_KEYR1</td>
<td>AES_KEYR2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0014</td>
<td>AES_KEYR1</td>
<td></td>
<td></td>
<td>AES_KEYR2</td>
<td>AES_KEYR3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0018</td>
<td>AES_KEYR2</td>
<td></td>
<td></td>
<td>AES_KEYR3</td>
<td>AES_KEYR4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0020</td>
<td>AES_IVR0</td>
<td></td>
<td></td>
<td>AES_KEYR4</td>
<td>AES_KEYR5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0024</td>
<td>AES_IVR1</td>
<td></td>
<td></td>
<td>AES_KEYR5</td>
<td>AES_KEYR6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0028</td>
<td>AES_IVR2</td>
<td></td>
<td></td>
<td>AES_KEYR6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x002C</td>
<td>AES_IVR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0030</td>
<td>AES_KEYR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0034</td>
<td>AES_KEYR5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0038</td>
<td>AES_KEYR6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Table 101. AES register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset Register</th>
<th>Offset Register</th>
<th>Offset Register</th>
<th>Offset Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>AES_CR</td>
<td>AES_SR</td>
<td>AES_DINR</td>
<td>AES_KEYR0</td>
<td>AES_KEYR1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0008</td>
<td>AES_SR</td>
<td></td>
<td>AES_DINR</td>
<td>AES_KEYR0</td>
<td>AES_KEYR1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x000C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0010</td>
<td>AES_KEYR0</td>
<td></td>
<td></td>
<td>AES_KEYR1</td>
<td>AES_KEYR2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0014</td>
<td>AES_KEYR1</td>
<td></td>
<td></td>
<td>AES_KEYR2</td>
<td>AES_KEYR3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0018</td>
<td>AES_KEYR2</td>
<td></td>
<td></td>
<td>AES_KEYR3</td>
<td>AES_KEYR4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0020</td>
<td>AES_IVR0</td>
<td></td>
<td></td>
<td>AES_KEYR4</td>
<td>AES_KEYR5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0024</td>
<td>AES_IVR1</td>
<td></td>
<td></td>
<td>AES_KEYR5</td>
<td>AES_KEYR6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0028</td>
<td>AES_IVR2</td>
<td></td>
<td></td>
<td>AES_KEYR6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x002C</td>
<td>AES_IVR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0030</td>
<td>AES_KEYR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0034</td>
<td>AES_KEYR5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0038</td>
<td>AES_KEYR6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Example register values

- `AES_CR`: Reset value $00000000000000000000000000000000$ meaning $00000000000000000000000000000000$
- `AES_SR`: Reset value $00000000000000000000000000000000$
- `AES_DINR`: Reset value $00000000000000000000000000000000$
- `AES_KEYR0`: Reset value $00000000000000000000000000000000$
- `AES_KEYR1`: Reset value $00000000000000000000000000000000$
- `AES_KEYR2`: Reset value $00000000000000000000000000000000$
- `AES_KEYR3`: Reset value $00000000000000000000000000000000$
- `AES_KEYR4`: Reset value $00000000000000000000000000000000$
- `AES_KEYR5`: Reset value $00000000000000000000000000000000$
- `AES_KEYR6`: Reset value $00000000000000000000000000000000$
### Table 101. AES register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x003</td>
<td>AES_KEYR7</td>
<td>KEY[255:224]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x0040</td>
<td>AES_SUSP0R</td>
<td>SUSP0[31:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x0044</td>
<td>AES_SUSP1R</td>
<td>SUSP1[31:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x0048</td>
<td>AES_SUSP2R</td>
<td>SUSP2[31:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x004C</td>
<td>AES_SUSP3R</td>
<td>SUSP3[31:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x0050</td>
<td>AES_SUSP4R</td>
<td>SUSP4[31:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x0054</td>
<td>AES_SUSP5R</td>
<td>SUSP5[31:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x0058</td>
<td>AES_SUSP6R</td>
<td>SUSP6[31:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x005C</td>
<td>AES_SUSP7R</td>
<td>SUSP7[31:0]</td>
<td>0x00000000</td>
</tr>
</tbody>
</table>

Refer to *Section 2.2 on page 54* for the register boundary addresses.
20 Advanced-control timers (TIM1)

In this section, “TIMx” should be understood as “TIM1” since there is only one instance of this type of timer for the products to which this reference manual applies.

20.1 TIM1 introduction

The advanced-control timer (TIM1) consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in Section 20.3.26: Timer synchronization.
20.2 TIM1 main features

TIM1 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 6 independent channels for:
  - Input Capture (but channels 5 and 6)
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management
1. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to Section 5.2.8:
     Clock security system (CSS)
   - A PVD output
   - SRAM parity error signal
   - Cortex®-M0+ LOCKUP (Hardfault) output.
   - COMPx output, x = 1, 2.
20.3 TIM1 functional description

20.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:
- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 104* and *Figure 105* give some examples of the counter behavior when the prescaler ratio is changed on the fly:
Figure 104. Counter timing diagram with prescaler division change from 1 to 2

Figure 105. Counter timing diagram with prescaler division change from 1 to 4
20.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):
- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.
Figure 106. Counter timing diagram, internal clock divided by 1

Figure 107. Counter timing diagram, internal clock divided by 2
**Figure 108. Counter timing diagram, internal clock divided by 4**

- **CK_PSC**
- **CNT_EN**
- **Timer clock = CK_CNT**
- **Counter register**: 0035 0036 0000 0001
- **Counter overflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**

**Figure 109. Counter timing diagram, internal clock divided by N**

- **CK_PSC**
- **Timer clock = CK_CNT**
- **Counter register**: 1F 20 00
- **Counter overflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**
**Figure 110. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)**

![Diagram](image1)

**Figure 111. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)**

![Diagram](image2)
**Downcounting mode**

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.
Figure 112. Counter timing diagram, internal clock divided by 1

Figure 113. Counter timing diagram, internal clock divided by 2
Figure 114. Counter timing diagram, internal clock divided by 4

Figure 115. Counter timing diagram, internal clock divided by N
Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

**Figure 117. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6**

1. Here, center-aligned mode 1 is used (for more details refer to Section 20.4: TIM1 registers).
Figure 118. Counter timing diagram, internal clock divided by 2

![Diagram](MS31190V1)

Figure 119. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

![Diagram](MS31191V1)

Note: Here, center_aligned mode 2 or 3 is updated with an UIF on overflow
Figure 120. Counter timing diagram, internal clock divided by N

Figure 121. Counter timing diagram, update event with ARPE=1 (counter underflow)
20.3.3 Repetition counter

Section 20.3.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode.

Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is 2xTck, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to Figure 123). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.
In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the overflow. If the RCR was written after launching the counter, the UEV occurs on the underflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

**Figure 123. Update rate examples depending on mode and TIMx_RCR register settings**

<table>
<thead>
<tr>
<th>Counter-aligned mode</th>
<th>Edge-aligned mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMx_RCR = 0</td>
<td>Upcounting</td>
</tr>
<tr>
<td>TIMx_RCR = 1</td>
<td>Downcounting</td>
</tr>
<tr>
<td>TIMx_RCR = 2</td>
<td>(by SW)</td>
</tr>
<tr>
<td>TIMx_RCR = 3</td>
<td>(by SW)</td>
</tr>
<tr>
<td>TIMx_RCR = 3 and re-synchronization</td>
<td>(by SW)</td>
</tr>
</tbody>
</table>

**UEV** → **Update event**: Preload registers transferred to active registers and update interrupt generated

**Update Event**: If the repetition counter underflow occurs when the counter is equal to the auto-reload value.
20.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see Section 20.3.5)
- trigger for the slave mode (see Section 20.3.26)
- PWM reset input for cycle-by-cycle current regulation (see Section 20.3.7)

Figure 124 below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield.

Figure 124. External trigger input block

The ETR input comes from multiple sources: input pins (default configuration), comparator outputs and analog watchdogs. The selection is done with the ETRSEL[3:0] bitfield.

Figure 125. TIM1 ETR input circuitry
20.3.5 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

**Internal clock source (CK_INT)**

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

*Figure 126* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

![Control circuit in normal mode, internal clock divided by 1](MS31085V2)

**External clock source mode 1**

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.
1. Codes ranging from 01000 to 11111 are reserved

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so the user does not need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The Figure 129 gives an overview of the external trigger input block.

For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. Refer to Figure 125: TIM1 ETR input circuitry.

For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:
1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register.
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register.
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

*Figure 130. Control circuit in external clock mode 2*
20.3.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 131 to Figure 134 give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 131. Capture/compare channel (example: channel 1 input stage)

The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.
Figure 132. Capture/compare channel 1 main circuit
Figure 133. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)

1. OCxREF, where x is the rank of the complementary channel

Figure 134. Output stage of capture/compare channel (channel 4)
1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

20.3.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when written with '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been
detected (sampled at $f_{DTS}$ frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

4. Select the edge of the active transition on the TI1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).

5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).

6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.

7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:
- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

20.3.8 PWM input mode

This mode is a particular case of input capture mode. The procedure is the same except:
- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, the user can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):
1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMRx register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMRx register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP=’10’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

**Figure 136. PWM input mode timing**

<table>
<thead>
<tr>
<th>TIMx_CNT</th>
<th>TIMx_CCR1</th>
<th>TIMx_CCR2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0004</td>
<td>0000</td>
<td>0001</td>
</tr>
<tr>
<td>0002</td>
<td>0003</td>
<td>0004</td>
</tr>
<tr>
<td>0000</td>
<td>0002</td>
<td></td>
</tr>
</tbody>
</table>

IC1 capture
IC2 capture
reset counter
IC2 capture
also with measurement
IC1 capture
period measurement

20.3.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCxREF/OCx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCxREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.
20.3.10 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while Channel 5 and 6 are only available inside the device (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM=0000), be set active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxCE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 137.
20.3.11 **PWM mode**

Pulse Width Modulation mode allows to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.
PWM edge-aligned mode

- **Upcounting configuration**
  
  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the Upcounting mode on page 489.

  In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxREF is held at ‘0’. Figure 138 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

  ![Figure 138. Edge-aligned PWM waveforms (ARR=8)](image)

- **Downcounting configuration**

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the Downcounting mode on page 493.

  In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF is held at ‘1’. 0% PWM is not possible in this mode.

PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the
TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to the *Center-aligned mode (up/down counting)* on page 496.

*Figure 139* shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

### Figure 139. Center-aligned PWM waveforms (ARR=8)

| Counter register | OCxREF | CCRx = 4 | CMS=01
CMS=10
CMS=11 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CCRx=7</td>
<td>OCxREF</td>
<td>CCxIF</td>
<td>CMS=10 or 11</td>
</tr>
</tbody>
</table>
| CCRx=8           | OCxREF | '1'      | CMS=01
CMS=10
CMS=11 |
| CCRx>8           | OCxREF | '1'      | CMS=01
CMS=10
CMS=11 |
| CCRx=0           | OCxREF | '0'      | CMS=01
CMS=10
CMS=11 |
| CCxIF            | CMS=01 | CMS=10   | CMS=11         |
| OCxREF           | CMS=01 | CMS=10   | CMS=11         |
| CCRx=4           | OCxREF | CCxIF    | CMS=10 or 11   |

**Hints on using center-aligned mode**

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit.
in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.

- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

### 20.3.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

**Note:** The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 1.

*Figure 140* represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.
20.3.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 141 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1.
20.3.14 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The OC5REF signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the OC5REF is combined. The resulting signals, OCxREFC, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, OC1REFC is controlled by TIMx_CCR1 and TIMx_CCR5
- If GC5C2 is set, OC2REFC is controlled by TIMx_CCR2 and TIMx_CCR5
- If GC5C3 is set, OC3REFC is controlled by TIMx_CCR3 and TIMx_CCR5

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].
20.3.15 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...).

The polarity of the outputs (main output OCx or complementary OCxN) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to Table 106: Output control bits for complementary OCx and OCxN channels with break feature on page 566 for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

The TRGO2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to Section 20.3.27: ADC synchronization for more details.
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

**Figure 143. Complementary output with dead-time insertion**

[Diagram showing OCxREF, OCx, and OCxN with delays indicated]

**Figure 144. Dead-time waveforms with delay greater than the negative pulse**

[Diagram showing OCxREF, OCx, and OCxN with delays indicated]
Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

20.3.16 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM1 timer. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.
The output enable signal and output levels during break are depending on several control bits:
- the MOE bit in TIMx_BDTR register allows to enable/disable the outputs by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to Table 106: Output control bits for complementary OCx and OCxN channels with break feature on page 566 for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKE/BK2E and BKP/BK2P can be modified at the same time. When the BKE/BK2E and BKP/BK2P bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_OR2 and TIMx_OR3 registers.

The sources for break (BRK) channel are:
- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source:
  - the Cortex®-M0+ LOCKUP output
  - the PVD output
  - the SRAM parity error signal
  - a flash ECC error
  - a clock failure event generated by the CSS detector
  - the output from a comparator, with polarity selection and optional digital filtering

The sources for break2 (BRK2) are:
- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source coming from a comparator output.

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register. The software break generation using BG and B2G is active whatever the BKE and BK2E enable bits values.
All sources are ORed before entering the timer BRK or BRK2 inputs, as per Figure 146 below.

**Figure 146. Break and Break2 circuitry overview**

Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.
When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSSl bit). This feature is enabled even if the MCU oscillator is off.

- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSl=0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.

- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 \(ck_{tim}\) clock cycles).
  - If OSSl=0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (SBIF, BIF and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set.

- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to ‘1’ again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

**Note:** The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows to freeze the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to Section 20.4.20: TIM1 break and dead-time register (TIM1_BDTR). The LOCK bits can be written only once after an MCU reset.

*Figure 147* shows an example of behavior of the outputs in response to a break.
Figure 147. Various output behavior in response to a break event on BRK (OSSI = 1)

- OCxREF
  - OCx (OCxN not implemented, CCxP=0, OISx=1)
  - OCx (OCxN not implemented, CCxP=0, OISx=0)
  - OCx (OCxN not implemented, CCxP=1, OISx=1)
  - OCx (OCxN not implemented, CCxP=1, OISx=0)

- OCxN
  - (CCxE=1, CCxP=0, OISx=0, CCxNE=1, CCxNP=0, OISx=0)
  - (CCxE=1, CCxP=0, OISx=1, CCxNE=1, CCxNP=1, OISx=1)
  - (CCxE=1, CCxP=0, OISx=0, CCxNE=0, CCxNP=0, OISx=0)
  - (CCxE=1, CCxP=0, CCxNE=0, CCxNP=0, OISx=OISx=0 or OISx=OISx=1)
The two break inputs have different behaviors on timer outputs:

- The BRK input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- BRK2 can only disable (inactive state) the PWM outputs.

The BRK has a higher priority than BRK2 input, as described in Table 102.

Note: **BRK2 must only be used with OSSR = OSSI = 1.**

### Table 102. Behavior of timer outputs versus BRK/BRK2 inputs

<table>
<thead>
<tr>
<th>BRK</th>
<th>BRK2</th>
<th>Timer outputs state</th>
<th>Typical use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>X</td>
<td></td>
<td>OCxN output (low side switches)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Inactive then forced output state (after a deadtime)</td>
<td>ON after deadtime insertion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Outputs disabled if OSSI = 0 (control taken over by GPIO logic)</td>
<td></td>
</tr>
<tr>
<td>Inactive</td>
<td>Active</td>
<td>Inactive</td>
<td>OCx output (high side switches)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OFF</td>
</tr>
</tbody>
</table>

*Figure 148* gives an example of OCx and OCxN output behavior in case of active signals on BRK and BRK2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

*Figure 148. PWM output state following BRK and BRK2 pins assertion (OSSI=1)*
20.3.17 Bidirectional break inputs

The TIM1 are featuring bidirectional break I/Os, as represented on Figure 150.

They allow to have:
- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break and break2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the break and break2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to ‘0’ to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BK(2)E = 1). When a software break event is generated with BK(2)E = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break(2) I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:
- The break input path is always active: a break event will be active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This will prevent the PWM output to be re-started as long as the break condition is present.
- The BK(2)DSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see Table 103)
Arming and re-arming break circuitry

The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

### Table 103. Break protection disarming conditions

<table>
<thead>
<tr>
<th>MOE</th>
<th>BKDIR (BK2DIR)</th>
<th>BKDSRM (BK2DSRM)</th>
<th>Break protection state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Disarmed</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Armed</td>
</tr>
</tbody>
</table>

**Figure 150. Output redirection (BRK2 request not represented)**
20.3.18 Clearing the OCxREF signal on an external event

The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode. ocref_clr_int input can be selected between the OCREF_CLR input and ETRF (ETR after the filter) by configuring the OCCS bit in the TIMx_SMCR register.

When ETRF is chosen, ETR must be configured as follows:

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to ‘00’.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to ‘0’.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

*Figure 151* shows the behavior of the OCxREF signal when the ETRF Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

*Note:* In case of a PWM with a 100% duty cycle (if CCRx>ARR), then OCxREF is enabled again at the next counter overflow.
### 20.3.19 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The Figure 152 describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

---

**Figure 152. 6-step generation, COM example (OSSR=1)**
### 20.3.20 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: \( CNT < CCRx \leq ARR \) (in particular, \( 0 < CCRx \))
- In downcounting: \( CNT > CCRx \)

#### Figure 153. Example of one pulse mode.

For example one may want to generate a positive pulse on OC1 with a length of \( t_{PULSE} \) and after a delay of \( t_{DELAY} \) as soon as a positive edge is detected on the TI2 input pin.

Let’s use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).
The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The $t_{\text{DELAY}}$ is defined by the value written in the TIMx_CCR1 register.
- The $t_{\text{PULSE}}$ is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let’s say one want to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0’ in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to ‘0’, so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay $t_{\text{DELAY}}$ min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 20.3.21 Retriggerable one pulse mode (OPM)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in Section 20.3.20:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

**Note:** The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.
20.3.22 Encoder interface mode

To select Encoder Interface mode write SMS='001' in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS='010' if it is counting on TI1 edges only and SMS='011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to a quadrature encoder. Refer to Table 104. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to ‘1’). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

Note: The prescaler must be set to zero when encoder mode is enabled

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder’s position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.
A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder’s differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The Figure 155 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx_CCMR2 register, TI1FP2 mapped on TI2).
- CC1P='0' and CC1NP='0' (TIMx_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='0' and CC2NP='0' (TIMx_CCER register, TI1FP2 non-inverted, TI1FP2=TI2).
- SMS='011' (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx_CR1 register, Counter enabled).

Table 104. Counting direction versus encoder signals

<table>
<thead>
<tr>
<th>Active edge</th>
<th>Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)</th>
<th>TI1FP1 signal</th>
<th>TI2FP2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counting on TI1 only</td>
<td>High</td>
<td>Down</td>
<td>Rising</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>Falling</td>
</tr>
<tr>
<td>Counting on TI2 only</td>
<td>High</td>
<td>No Count</td>
<td>No Count</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>No Count</td>
<td>No Count</td>
</tr>
<tr>
<td>Counting on TI1 and TI2</td>
<td>High</td>
<td>Down</td>
<td>Rising</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>Falling</td>
</tr>
</tbody>
</table>

Figure 155. Example of counter operation in encoder interface mode.
Figure 156 gives an example of counter behavior when Ti1FP1 polarity is inverted (same configuration as above except CC1P=’1’).

Figure 156. Example of encoder interface mode with Ti1FP1 polarity inverted.

The timer, when configured in Encoder Interface mode provides information on the sensor’s current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register’s bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter’s most significant bit is only accessible in write mode).

20.3.23 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register’s bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the UIF and UIFCPY flags assertion.
20.3.24 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per Figure 157 below.

![Figure 157. Measuring time interval between edges on 3 signals](image)

20.3.25 Interfacing with Hall sensors

This is done using the advanced-control timer (TIM1) to generate PWM signals to drive the motor and another timer TIMx (TIM2, TIM3) referred to as “interfacing timer” in Figure 158.

The “interfacing timer” captures the 3 timer input pins (CC1, CC2, CC3) connected through a XOR to the TI1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is TI1F_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TRC (See Figure 131: Capture/compare channel (example: channel 1 input stage) on page 507). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1) through the TRGO output.
Example: one wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’,
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx_CCMR1 register to ‘01’. The digital filter can also be programmed if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the TIMx_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx_CR2 register to ‘101’,

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

The Figure 158 describes this example.
Figure 158. Example of Hall sensor interface

Write CCxE, CCxNE and OCxM for next step
20.3.26  Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. They can be synchronized in several modes: Reset mode, Gated mode, and Trigger mode.

**Slave mode: Reset mode**

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

**Figure 159. Control circuit in reset mode**

![Control circuit in reset mode](image)
Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 160. Control circuit in Gated mode

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1.
register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 161. Control circuit in trigger mode

Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:
1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.

2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01 in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP=’0’ in TIMx_CCER register to validate the polarity (and detect rising edge only).

3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

**Figure 162. Control circuit in external clock mode 2 + trigger mode**

**Note:** The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.
20.3.27 ADC synchronization

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the TRGO2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in Figure 142 on page 520.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

20.3.28 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address, i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.
This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.

2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:
   DBL = 3 transfers, DBA = 0xE.

3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).

4. Enable TIMx

5. Enable the DMA channel

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

*Note:* A null value can be written to the reserved registers.

### 20.3.29 Debug mode

When the microcontroller enters debug mode (Cortex®-M0+ core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module.

For safety purposes, when the counter is stopped (DBG_TIMx_STOP = 1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to *Section 37.10.4: DBG APB freeze register 2 (DBG_APB_FZ2)*. For safety purposes, when the counter is stopped (DBG_TIMx_STOP = 1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.
20.4 **TIM1 registers**

Refer to for a list of abbreviations used in register descriptions.

20.4.1 **TIM1 control register 1 (TIM1_CR1)**

Address offset: 0x00

Reset value: 0x0000

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

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (tDTS) used by the dead-time generators and the digital filters (ETR, T1x),

00: tDTS=CK_INT
01: tDTS=2*CK_INT
10: tDTS=4*CK_INT
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

**Note**: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

**Note**: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.
Bit 3 **OPM**: One pulse mode
   0: Counter is not stopped at update event
   1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
   This bit is set and cleared by software to select the UEV event sources.
   0: Any of the following events generate an update interrupt or DMA request if enabled.
   These events can be:
   - Counter overflow/underflow
   - Setting the UG bit
   - Update generation through the slave mode controller
   1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
   This bit is set and cleared by software to enable/disable UEV event generation.
   0: UEV enabled. The Update (UEV) event is generated by one of the following events:
   - Counter overflow/underflow
   - Setting the UG bit
   - Update generation through the slave mode controller
   Buffered registers are then loaded with their preload values.
   1: UEV disabled. The Update event is not generated, shadow registers keep their value
   (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is
   set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
   0: Counter disabled
   1: Counter enabled

   **Note**: **External clock, gated mode and encoder mode can work only if the CEN bit has been**
   **previously set by software. However trigger mode can set the CEN bit automatically by**
   **hardware.**

### 20.4.2 TIM1 control register 2 (TIM1_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:24  Reserved, must be kept at reset value.

Bits 23:20  **MMS2[3:0]: Master mode selection 2**

These bits allow the information to be sent to ADC for synchronization (TRGO2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on TRGO2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - the update event is selected as trigger output (TRGO2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (TRGO2).

0100: **Compare** - OC1REF signal is used as trigger output (TRGO2)

0101: **Compare** - OC2REF signal is used as trigger output (TRGO2)

0110: **Compare** - OC3REF signal is used as trigger output (TRGO2)

0111: **Compare** - OC4REF signal is used as trigger output (TRGO2)

1000: **Compare** - OC5REF signal is used as trigger output (TRGO2)

1001: **Compare** - OC6REF signal is used as trigger output (TRGO2)

1010: **Compare Pulse** - OC4REF rising or falling edges generate pulses on TRGO2

1011: **Compare Pulse** - OC6REF rising or falling edges generate pulses on TRGO2

1100: **Compare Pulse** - OC4REF or OC6REF rising edges generate pulses on TRGO2

1101: **Compare Pulse** - OC4REF rising or OC6REF falling edges generate pulses on TRGO2

1110: **Compare Pulse** - OC5REF or OC6REF rising edges generate pulses on TRGO2

1111: **Compare Pulse** - OC5REF rising or OC6REF falling edges generate pulses on TRGO2

*Note:* The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19  Reserved, must be kept at reset value.

Bit 18  **OIS6**: Output Idle state 6 (OC6 output)

Refer to OIS1 bit

Bit 17  Reserved, must be kept at reset value.

Bit 16  **OIS5**: Output Idle state 5 (OC5 output)

Refer to OIS1 bit

Bit 15  Reserved, must be kept at reset value.

Bit 14  **OIS4**: Output Idle state 4 (OC4 output)

Refer to OIS1 bit

Bit 13  **OIS3N**: Output Idle state 3 (OC3N output)

Refer to OIS1N bit

Bit 12  **OIS3**: Output Idle state 3 (OC3 output)

Refer to OIS1 bit
Bit 11 **OIS2N**: Output Idle state 2 (OC2N output)  
Refer to OIS1N bit

Bit 10 **OIS2**: Output Idle state 2 (OC2 output)  
Refer to OIS1 bit

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)  
0: OC1N=0 after a dead-time when MOE=0  
1: OC1N=1 after a dead-time when MOE=0  
*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)  
0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0  
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0  
*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 7 **TI1S**: TI1 selection  
0: The TIMx_CH1 pin is connected to TI1 input  
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection  
These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:  
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.  
001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).  
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.  
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO)  
100: **Compare** - OC1REF signal is used as trigger output (TRGO)  
101: **Compare** - OC2REF signal is used as trigger output (TRGO)  
110: **Compare** - OC3REF signal is used as trigger output (TRGO)  
111: **Compare** - OC4REF signal is used as trigger output (TRGO)  
*Note: The clock of the slave timer or ADC must be enabledprior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.*

Bit 3 **CCDS**: Capture/compare DMA selection  
0: CCx DMA request sent when CCx event occurs  
1: CCx DMA requests sent when update event occurs
Bit 2 **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI

*Note:* This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).

*Note:* This bit acts only on channels that have a complementary output.

### 20.4.3 TIM1 slave mode control register (TIM1_SMCR)

**Address offset:** 0x08

**Reset value:** 0x0000 0000

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

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

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity
This bit selects whether ETR or ETR is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge.
1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable
This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

*Note:* 1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).
2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).
3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.
Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

- 00: Prescaler OFF
- 01: ETRP frequency divided by 2
- 10: ETRP frequency divided by 4
- 11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at fDTS
- 0001: fSAMPLING=fCK_INT, N=2
- 0010: fSAMPLING=fCK_INT, N=4
- 0011: fSAMPLING=fCK_INT, N=8
- 0100: fSAMPLING=fDTS/2, N=6
- 0101: fSAMPLING=fDTS/2, N=8
- 0110: fSAMPLING=fDTS/4, N=6
- 0111: fSAMPLING=fDTS/4, N=8
- 1000: fSAMPLING=fDTS/8, N=6
- 1001: fSAMPLING=fDTS/8, N=8
- 1010: fSAMPLING=fDTS/16, N=5
- 1011: fSAMPLING=fDTS/16, N=6
- 1100: fSAMPLING=fDTS/32, N=5
- 1101: fSAMPLING=fDTS/32, N=6
- 1110: fSAMPLING=fDTS/32, N=8
- 1111: fSAMPLING=fDTS/32, N=8

Bit 7 **MSM**: Master/slave mode

- 0: No action
- 1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

- 00000: Internal Trigger 0 (ITR0)
- 00001: Internal Trigger 1 (ITR1)
- 00010: Internal Trigger 2 (ITR2)
- 00011: Internal Trigger 3 (ITR3)
- 00100: TI1 Edge Detector (TI1F_ED)
- 00101: Filtered Timer Input 1 (TI1FP1)
- 00110: Filtered Timer Input 2 (TI2FP2)
- 00111: External Trigger input (ETRF)

Others: Reserved

See Table 105: TIM1 internal trigger connection on page 552 for more details on ITRx meaning for each Timer.

**Note:** These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

**Note:** The other bit is at position 16 in the same register
Bit 3 **OCCS**: OCREF clear selection
This bit is used to select the OCREF clear source.
- 0: OCREF_CLR_INT is connected to COMP1 or COMP2 output depending on TIM1_OR1.OCREF_CLR
- 1: OCREF_CLR_INT is connected to ETRF

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection
When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.
- 0000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.
- 0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
- 0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
- 0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
- 0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
- 0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
- 0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
- 0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
- 1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.
- Codes above 1000: Reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=00100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 105. TIM1 internal trigger connection

<table>
<thead>
<tr>
<th>Slave TIM</th>
<th>ITR0 (TS = 00000)</th>
<th>ITR1 (TS = 00001)</th>
<th>ITR2 (TS = 00010)</th>
<th>ITR3 (TS = 00011)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM1</td>
<td>TIM15</td>
<td>TIM2</td>
<td>TIM3</td>
<td>TIM17 OC1</td>
</tr>
</tbody>
</table>

20.4.4 **TIM1 DMA/interrupt enable register (TIM1_DIER)**
Address offset: 0x0C
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>TDE</td>
<td>COMDE</td>
<td>CC4DE</td>
<td>CC3DE</td>
<td>CC2DE</td>
<td>CC1DE</td>
<td>UDE</td>
<td>BIE</td>
<td>TIE</td>
<td>COMIE</td>
<td>CC4IE</td>
<td>CC3IE</td>
<td>CC2IE</td>
<td>CC1IE</td>
<td>UIE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>Bit 15</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 14</td>
<td><strong>TDE</strong>: Trigger DMA request enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Trigger DMA request disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Trigger DMA request enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 13</td>
<td><strong>COMDE</strong>: COM DMA request enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: COM DMA request disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: COM DMA request enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 12</td>
<td><strong>CC4DE</strong>: Capture/Compare 4 DMA request enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: CC4 DMA request disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: CC4 DMA request enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 11</td>
<td><strong>CC3DE</strong>: Capture/Compare 3 DMA request enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: CC3 DMA request disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: CC3 DMA request enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 10</td>
<td><strong>CC2DE</strong>: Capture/Compare 2 DMA request enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: CC2 DMA request disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: CC2 DMA request enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 9</td>
<td><strong>CC1DE</strong>: Capture/Compare 1 DMA request enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: CC1 DMA request disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: CC1 DMA request enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 8</td>
<td><strong>UDE</strong>: Update DMA request enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Update DMA request disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Update DMA request enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 7</td>
<td><strong>BIE</strong>: Break interrupt enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Break interrupt disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Break interrupt enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 6</td>
<td><strong>TIE</strong>: Trigger interrupt enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Trigger interrupt disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Trigger interrupt enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 5</td>
<td><strong>COMIE</strong>: COM interrupt enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: COM interrupt disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: COM interrupt enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 4</td>
<td><strong>CC4IE</strong>: Capture/Compare 4 interrupt enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: CC4 interrupt disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: CC4 interrupt enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 3</td>
<td><strong>CC3IE</strong>: Capture/Compare 3 interrupt enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: CC3 interrupt disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: CC3 interrupt enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
20.4.5 TIM1 status register (TIM1_SR)

Address offset: 0x10
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-18</td>
<td>Reserved, must be kept at reset value.</td>
<td>rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0</td>
</tr>
<tr>
<td>17</td>
<td>CC6IF: Compare 6 interrupt flag</td>
<td>Refer to CC1IF description (Note: Channel 6 can only be configured as output)</td>
</tr>
<tr>
<td>16</td>
<td>CC5IF: Compare 5 interrupt flag</td>
<td>Refer to CC1IF description (Note: Channel 5 can only be configured as output)</td>
</tr>
<tr>
<td>15-14</td>
<td>Reserved, must be kept at reset value.</td>
<td>rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0 rc_w0</td>
</tr>
<tr>
<td>13</td>
<td>SBIF: System Break interrupt flag</td>
<td>This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active. This flag must be reset to re-start PWM operation.</td>
</tr>
<tr>
<td>12</td>
<td>CC4OF: Capture/Compare 4 overcapture flag</td>
<td>Refer to CC1OF description</td>
</tr>
<tr>
<td>11</td>
<td>CC3OF: Capture/Compare 3 overcapture flag</td>
<td>Refer to CC1OF description</td>
</tr>
<tr>
<td>10</td>
<td>CC2OF: Capture/Compare 2 overcapture flag</td>
<td>Refer to CC1OF description</td>
</tr>
</tbody>
</table>
Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag
This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.
0: No break event occurred.
1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred.
1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag
This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.
0: No COM event occurred.
1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag
**If channel CC1 is configured as output:** This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

**If channel CC1 is configured as input:** This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred
1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)
Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by a trigger event (refer to Section 20.4.3: TIM1 slave mode control register (TIM1_SMCR)), if URS=0 and UDIS=0 in the TIMx_CR1 register.

### 20.4.6 TIM1 event generation register (TIM1_EGR)

Address offset: 0x14

Reset value: 0x0000

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

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

Bit 8 **B2G**: Break 2 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action
1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action
1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits

*Note: This bit acts only on channels having a complementary output.*

Bit 4 **CC4G**: Capture/Compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation

Refer to CC1G description
Bit 2  **CC2G**: Capture/Compare 2 generation  
Refer to CC1G description

Bit 1  **CC1G**: Capture/Compare 1 generation  
This bit is set by software in order to generate an event, it is automatically cleared by hardware.  
0: No action  
1: A capture/compare event is generated on channel 1:  
If channel CC1 is configured as output:  
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.  
If channel CC1 is configured as input:  
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set,  
the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the  
CC1IF flag was already high.

Bit 0  **UG**: Update generation  
This bit can be set by software, it is automatically cleared by hardware.  
0: No action  
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler  
counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if  
the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload  
value (TIMx_ARR) if DIR=1 (downcounting).

### 20.4.7 TIM1 capture/compare mode register 1 [alternate]  
(TIM1_CCMR1)

Address offset: 0x18  
Reset value: 0x0000 0000  
The same register can be used for input capture mode (this section) or for output compare  
mode (next section). The direction of a channel is defined by configuring the corresponding  
CCxS bits. All the other bits of this register have a different function for input capture and for  
output compare modes. It is possible to combine both modes independently (e.g. channel 1  
in input capture mode and channel 2 in output compare mode).

**Input capture mode:**

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

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

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:12  **IC2F[3:0]**: Input capture 2 filter  
Refer to IC1F[3:0] description.

Bits 11:10  **IC2PSC[1:0]**: Input capture 2 prescaler  
Refer to OC1PSC[1:0] description.
Bits 9:8  **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- 00: CC2 channel is configured as output
- 01: CC2 channel is configured as input, IC2 is mapped on TI2
- 10: CC2 channel is configured as input, IC2 is mapped on TI1
- 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).*

Bits 7:4  **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at fDTS
- 0010: fSAMPLING=fCK_INT, N=4
- 0011: fSAMPLING=fCK_INT, N=8
- 0100: fSAMPLING=fDTS/2, N=6
- 0101: fSAMPLING=fDTS/2, N=8
- 1100: fSAMPLING=fDTS/8, N=8
- 1101: fSAMPLING=fDTS/8, N=8
- 1110: fSAMPLING=fDTS/16, N=6
- 1111: fSAMPLING=fDTS/16, N=8

Bits 3:2  **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0  **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, IC1 is mapped on TI1
- 10: CC1 channel is configured as input, IC1 is mapped on TI2
- 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).*

### 20.4.8 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)

**Address offset:** 0x18

**Reset value:** 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the
corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

**Output compare mode:**

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **OC2M[3]:** Output Compare 2 mode - bit 3
Refer to OC2M description on bits 14:12.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **OC1M[3]:** Output Compare 1 mode - bit 3
Refer to OC1M description on bits 6:4

Bit 15 **OC2CE:** Output Compare 2 clear enable
Refer to OC1CE description.

Bits 14:12 **OC2M[2:0]:** Output Compare 2 mode
Refer to OC1M[2:0] description.

Bit 11 **OC2PE:** Output Compare 2 preload enable
Refer to OC1PE description.

Bit 10 **OC2FE:** Output Compare 2 fast enable
Refer to OC1FE description.

Bits 9:8 **CC2S[1:0]:** Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

**Note:** **CC2S bits are writable only when the channel is OFF (CC2E = ‘0’ in TIMx_CCER).**

Bit 7 **OC1CE:** Output Compare 1 clear enable
0: OC1Ref is not affected by the ocref_clr_int signal
1: OC1Ref is cleared as soon as a High level is detected on ocref_clr_int signal (OCREFCLR input or ETRF input)
Bits 6:4 **OC1M[2:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. (This mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0') as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF='1').

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggernable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggernable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

**Note:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

**Note:** In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

**Note:** On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.
Bit 3 OC1PE: Output Compare 1 preload enable
0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.
Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).
2: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 OC1FE: Output Compare 1 fast enable
This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 CC1S[1:0]: Capture/Compare 1 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

20.4.9 TIM1 capture/compare mode register 2 [alternate]
(TIM1_CCMR2)

Address offset: 0x1C
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Input capture mode:
Bits 31:16  Reserved, must be kept at reset value.

Bits 15:12  **IC4F[3:0]**: Input capture 4 filter
Refer to **IC1F[3:0]** description.

Bits 11:10  **IC4PSC[1:0]**: Input capture 4 prescaler
Refer to **IC1PSC[1:0]** description.

Bits 9:8  **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).*

Bits 7:4  **IC3F[3:0]**: Input capture 3 filter
Refer to **IC1F[3:0]** description.

Bits 3:2  **IC3PSC[1:0]**: Input capture 3 prescaler
Refer to **IC1PSC[1:0]** description.

Bits 1:0  **CC3S[1:0]**: Capture/compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).*

---

### 20.4.10 TIM1 capture/compare mode register 2 [alternate]
*(TIM1_CCMR2)*

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

#### Output compare mode

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:25  Reserved, must be kept at reset value.

Bit 24  **OC4M[3]**: Output Compare 4 mode - bit 3
Refer to OC1M description.

Bits 23:17  Reserved, must be kept at reset value.

Bit 16  **OC3M[3]**: Output Compare 3 mode - bit 3
Refer to OC1M description.

Bit 15  **OC4CE**: Output compare 4 clear enable
Refer to OC1CE description.

Bits 14:12  **OC4M[2:0]**: Output compare 4 mode
Refer to OC4M description.

Bit 11  **OC4PE**: Output compare 4 preload enable
Refer to OC1PE description.

Bit 10  **OC4FE**: Output compare 4 fast enable
Refer to OC1FE description.

Bits 9:8  **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note:* **CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).**

Bit 7  **OC3CE**: Output compare 3 clear enable
Refer to OC1CE description.

Bits 6:4  **OC3M[2:0]**: Output compare 3 mode
Refer to OC1M description.

Bit 3  **OC3PE**: Output compare 3 preload enable
Refer to OC1PE description.

Bit 2  **OC3FE**: Output compare 3 fast enable
Refer to OC1FE description.

Bits 1:0  **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note:* **CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).**
20.4.11 TIM1 capture/compare enable register (TIM1_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

| Bit 31:22 | Reserved, must be kept at reset value. |
| Bit 21 | **CC6P**: Capture/Compare 6 output polarity  
Refer to CC1P description |
| Bit 20 | **CC6E**: Capture/Compare 6 output enable  
Refer to CC1E description |
| Bit 19:18 | Reserved, must be kept at reset value. |
| Bit 17 | **CC5P**: Capture/Compare 5 output polarity  
Refer to CC1P description |
| Bit 16 | **CC5E**: Capture/Compare 5 output enable  
Refer to CC1E description |
| Bit 15 | **CC4NP**: Capture/Compare 4 complementary output polarity  
Refer to CC1NP description |
| Bit 14 | Reserved, must be kept at reset value. |
| Bit 13 | **CC4P**: Capture/Compare 4 output polarity  
Refer to CC1P description |
| Bit 12 | **CC4E**: Capture/Compare 4 output enable  
Refer to CC1E description |
| Bit 11 | **CC3NP**: Capture/Compare 3 complementary output polarity  
Refer to CC1NP description |
| Bit 10 | **CC3NE**: Capture/Compare 3 complementary output enable  
Refer to CC1NE description |
| Bit 9 | **CC3P**: Capture/Compare 3 output polarity  
Refer to CC1P description |
| Bit 8 | **CC3E**: Capture/Compare 3 output enable  
Refer to CC1E description |
| Bit 7 | **CC2NP**: Capture/Compare 2 complementary output polarity  
Refer to CC1NP description |
| Bit 6 | **CC2NE**: Capture/Compare 2 complementary output enable  
Refer to CC1NE description |
Bit 5  **CC2P**: Capture/Compare 2 output polarity  
Refer to CC1P description

Bit 4  **CC2E**: Capture/Compare 2 output enable  
Refer to CC1E description

Bit 3  **CC1NP**: Capture/Compare 1 complementary output polarity  
**CC1 channel configured as output**:  
0: OC1N active high.  
1: OC1N active low.  
**CC1 channel configured as input**:  
This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.  
*Note*: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (channel configured as output).  
On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2  **CC1NE**: Capture/Compare 1 complementary output enable  
0: Off - OC1N is not active. OC1N level is then function of MOE, OSS1, OSS1R, OIS1, OIS1N and CC1E bits.  
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSS1, OSS1R, OIS1, OIS1N and CC1E bits.  
On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1  **CC1P**: Capture/Compare 1 output polarity  
**CC1 channel configured as output**:  
0: OC1 active high  
1: OC1 active low  
**CC1 channel configured as input**: CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.  
00: non-inverted/rising edge. The circuit is sensitive to Ti1FP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), Ti1FP1 is not inverted (trigger operation in gated mode or encoder mode).  
01: inverted/falling edge. The circuit is sensitive to Ti1FP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), Ti1FP1 is inverted (trigger operation in gated mode or encoder mode).  
10: reserved, do not use this configuration.  
11: non-inverted/both edges/ The circuit is sensitive to both Ti1FP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), Ti1FP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.  
*Note*: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).  
On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.
Bit 0  **CC1E**: Capture/Compare 1 output enable

**CC1 channel configured as output:**
0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

**CC1 channel configured as input:** This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled.
1: Capture enabled.

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

<table>
<thead>
<tr>
<th>Control bits</th>
<th>Output states(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOE bit</td>
<td>OSSI bit</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

**Note:** BRK2 can only be used if OSSI = OSSR = 1.
Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.

20.4.12 TIM1 counter (TIM1_CNT)
Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UIF CPY: UIF copy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.</td>
</tr>
</tbody>
</table>

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 CNT[15:0]: Counter value

20.4.13 TIM1 prescaler (TIM1_PSC)
Address offset: 0x28
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>PSC[15:0]: Prescaler value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).</td>
<td></td>
</tr>
<tr>
<td>PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).</td>
<td></td>
</tr>
</tbody>
</table>

20.4.14 TIM1 auto-reload register (TIM1_ARR)
Address offset: 0x2C
Reset value: 0xFFFF

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>ARR[15:0]:</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit contains the value to be compared to the CNT to set the UIF bit of the TIMx_ISR register.</td>
<td></td>
</tr>
<tr>
<td>The counter stops when the CNT value matches ARR.</td>
<td></td>
</tr>
<tr>
<td>The counter is cleared when UG flag is set in TIMx_EGR register or through trigger controller when configured in “reset mode”).</td>
<td></td>
</tr>
</tbody>
</table>
Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register. Refer to the Section 20.3.1: Time-base unit on page 487 for more details about ARR update and behavior. The counter is blocked while the auto-reload value is null.

**20.4.15 TIM1 repetition counter register (TIM1_RCR)**

Address offset: 0x30

Reset value: 0x0000

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

Bits 15:0 **REP[15:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is loaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:
- the number of PWM periods in edge-aligned mode
- the number of half PWM period in center-aligned mode.

**20.4.16 TIM1 capture/compare register 1 (TIM1_CCR1)**

Address offset: 0x34

Reset value: 0x0000

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

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.
20.4.17  TIM1 capture/compare register 2 (TIM1_CCR2)

Address offset: 0x38
Reset value: 0x0000

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

Bits 15:0  **CCR2[15:0]: Capture/Compare 2 value**

- **If channel CC2 is configured as output:** CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
  
  It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
  
  The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output.

- **If channel CC2 is configured as input:** CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

20.4.18  TIM1 capture/compare register 3 (TIM1_CCR3)

Address offset: 0x3C
Reset value: 0x0000

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

Bits 15:0  **CCR3[15:0]: Capture/Compare value**

- **If channel CC3 is configured as output:** CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
  
  It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
  
  The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC3 output.

- **If channel CC3 is configured as input:** CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.
20.4.19  TIM1 capture/compare register 4 (TIM1_CCR4)

Address offset: 0x40
Reset value: 0x0000

<table>
<thead>
<tr>
<th>CCR4[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:0  **CCR4[15:0]: Capture/Compare value**

- **If channel CC4 is configured as output:** CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).
- It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
- The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.

- **If channel CC4 is configured as input:** CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

20.4.20  TIM1 break and dead-time register

(TIM1_BDTR)

Address offset: 0x44
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MOE</th>
<th>AOE</th>
<th>BKP</th>
<th>BKE</th>
<th>OSSR</th>
<th>OSSI</th>
<th>LOCK[1:0]</th>
<th>DTG[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Note:** As the bits BK2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

- **Bits 31:30 Reserved, must be kept at reset value.**
- **Bit 29  BK2BID:** Break2 bidirectional
  - Refer to BKBID description
Bit 28  **BKBID**: Break Bidirectional  
0: Break input BRK in input mode  
1: Break input BRK in bidirectional mode  
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.  
*Note*: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).  
*Note*: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.  

Bit 27  **BK2DSRM**: Break2 Disarm  
Refer to BKDSRM description  

Bit 26  **BKDSRM**: Break Disarm  
0: Break input BRK is armed  
1: Break input BRK is disarmed  
This bit is cleared by hardware when no break source is active.  
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.  
*Note*: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.  

Bit 25  **BK2P**: Break 2 polarity  
0: Break input BRK2 is active low  
1: Break input BRK2 is active high  
*Note*: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).  
*Note*: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.  

Bit 24  **BK2E**: Break 2 enable  
0: Break input BRK2 disabled  
1: Break input BRK2 enabled  
*Note*: The BRK2 must only be used with OSSR = OSSI = 1.  
*Note*: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).  
*Note*: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
Bits 23:20  BK2F[3:0]: Break 2 filter

This bit-field defines the frequency used to sample BRK2 input and the length of the digital filter applied to BRK2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, BRK2 acts asynchronously
- 0001: \( f_{SAMPLING} = f_{CK\_INT} \), \( N=2 \)
- 0010: \( f_{SAMPLING} = f_{CK\_INT} \), \( N=4 \)
- 0011: \( f_{SAMPLING} = f_{DTS}/2 \), \( N=8 \)
- 0100: \( f_{SAMPLING} = f_{DTS}/2 \), \( N=6 \)
- 0101: \( f_{SAMPLING} = f_{DTS}/2 \), \( N=8 \)
- 0110: \( f_{SAMPLING} = f_{DTS}/4 \), \( N=6 \)
- 0111: \( f_{SAMPLING} = f_{DTS}/4 \), \( N=8 \)
- 1000: \( f_{SAMPLING} = f_{DTS}/8 \), \( N=6 \)
- 1001: \( f_{SAMPLING} = f_{DTS}/8 \), \( N=8 \)
- 1010: \( f_{SAMPLING} = f_{DTS}/16 \), \( N=5 \)
- 1011: \( f_{SAMPLING} = f_{DTS}/16 \), \( N=6 \)
- 1100: \( f_{SAMPLING} = f_{DTS}/16 \), \( N=8 \)
- 1101: \( f_{SAMPLING} = f_{DTS}/32 \), \( N=5 \)
- 1110: \( f_{SAMPLING} = f_{DTS}/32 \), \( N=6 \)
- 1111: \( f_{SAMPLING} = f_{DTS}/32 \), \( N=8 \)

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 19:16  BKF[3:0]: Break filter

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, BRK acts asynchronously
- 0001: \( f_{SAMPLING} = f_{CK\_INT} \), \( N=2 \)
- 0010: \( f_{SAMPLING} = f_{CK\_INT} \), \( N=4 \)
- 0011: \( f_{SAMPLING} = f_{DTS}/2 \), \( N=8 \)
- 0100: \( f_{SAMPLING} = f_{DTS}/2 \), \( N=6 \)
- 0101: \( f_{SAMPLING} = f_{DTS}/2 \), \( N=8 \)
- 0110: \( f_{SAMPLING} = f_{DTS}/4 \), \( N=6 \)
- 0111: \( f_{SAMPLING} = f_{DTS}/4 \), \( N=8 \)
- 1000: \( f_{SAMPLING} = f_{DTS}/8 \), \( N=6 \)
- 1001: \( f_{SAMPLING} = f_{DTS}/8 \), \( N=8 \)
- 1010: \( f_{SAMPLING} = f_{DTS}/16 \), \( N=5 \)
- 1011: \( f_{SAMPLING} = f_{DTS}/16 \), \( N=6 \)
- 1100: \( f_{SAMPLING} = f_{DTS}/16 \), \( N=8 \)
- 1101: \( f_{SAMPLING} = f_{DTS}/32 \), \( N=5 \)
- 1110: \( f_{SAMPLING} = f_{DTS}/32 \), \( N=6 \)
- 1111: \( f_{SAMPLING} = f_{DTS}/32 \), \( N=8 \)

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 15 MOE: Main output enable
This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (BRK or BRK2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.
0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).
See OC/OCN enable description for more details (Section 20.4.11: TIM1 capture/compare enable register (TIM1 CCER)).

Bit 14 AOE: Automatic output enable
0: MOE can be set only by software
1: MOE can be set by software or automatically at the next update event (if none of the break inputs BRK and BRK2 is active)
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 BKP: Break polarity
0: Break input BRK is active low
1: Break input BRK is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 BKE: Break enable
This bit enables the complete break protection (including all sources connected to bk_acth and BKIN sources, as per Figure 146: Break and Break2 circuitry overview).
0: Break function disabled
1: Break function enabled
Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 OSSR: Off-state selection for Run mode
This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.
See OC/OCN enable description for more details (Section 20.4.11: TIM1 capture/compare enable register (TIM1 CCER)).
0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).
1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).
Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).
**Advanced-control timers (TIM1)**

**Bit 10 OSSI: Off-state selection for Idle mode**

This bit is used when MOE=0 due to a break event or by a software write, on channels configured as outputs.

See OC/OCN enable description for more details ([Section 20.4.11: TIM1 capture/compare enable register (TIM1_CCER)]).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

**Note:** This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

**Bits 9:8 LOCK[1:0]: Lock configuration**

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BK2SID, BK4BID, BK2DSRM, BK4DSRM, BK2P, BK2E, BK2F[3:0], BK4F[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

**Note:** The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

**Bits 7:0 DTG[7:0]: Dead-time generator setup**

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x tDTS with tDTS=TdTS.

DTG[7:5]=10x => DT=(64+DTG[5:0])x tDTS with tDTS=2xTdTS.

DTG[7:5]=110 => DT=(32+DTG[4:0])x tDTS with tDTS=8xTdTS.

DTG[7:5]=111 => DT=(32+DTG[4:0])x tDTS with tDTS=16xTdTS.

Example if TdTS=125ns (8MHz), dead-time possible values are:

- 0 to 15875 ns by 125 ns steps,
- 16 us to 31750 ns by 250 ns steps,
- 32 us to 63us by 1 us steps,
- 64 us to 126 us by 2 us steps

**Note:** This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

### 20.4.21 TIM1 DMA control register (TIM1_DCR)

**Address offset:** 0x48

**Reset value:** 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.
**Bits 12:8 DBL[4:0]: DMA burst length**

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

- 00000: 1 transfer
- 00001: 2 transfers
- 00010: 3 transfers
- ...
- 10001: 18 transfers

**Example:** Let us consider the following transfer: DBL = 7 bytes & DBA = TIMx_CR1.

- If DBL = 7 bytes and DBA = TIMx_CR1 represents the address of the byte to be transferred, the address of the transfer should be given by the following equation:

  $$(\text{TIMx_CR1 address}) + \text{DBA} + (\text{DMA index})$$

  In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data will be copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

  According to the configuration of the DMA Data Size, several cases may occur:

  - If the DMA Data Size is configured in half-words, 16-bit data will be transferred to each of the 7 registers.
  - If the DMA Data Size is configured in bytes, the data will also be transferred to 7 registers: the first register will contain the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

**Bits 7:5** Reserved, must be kept at reset value.

**Bits 4:0 DBA[4:0]: DMA base address**

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

**Example:**

- 00000: TIMx_CR1,
- 00001: TIMx_CR2,
- 00010: TIMx_SMCR,
- ...

### 20.4.22 TIM1 DMA address for full transfer (TIM1_DMAR)

**Address offset:** 0x4C

**Reset value:** 0x0000 0000

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

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

---

**ST**
Advanced-control timers (TIM1)  

20.4.23 TIM1 option register 1 (TIM1_OR1)

Address offset: 0x50
Reset value: 0x0000 0000

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

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

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

Bit 0 **OCREF_CLR**: Ocref_clr source selection
This bit selects the ocref_clr input source.
0: COMP1 output is connected to the OCREF_CLR input
1: COMP2 output is connected to the OCREF_CLR input

20.4.24 TIM1 capture/compare mode register 3  (TIM1_CCMR3)

Address offset: 0x54
Reset value: 0x0000 0000

The channels 5 and 6 can only be configured in output.

**Output compare mode:**

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

<table>
<thead>
<tr>
<th>OC6 CE</th>
<th>OC6M[2:0]</th>
<th>OC6 PE</th>
<th>OC6FE</th>
<th>OC5 CE</th>
<th>OC5M[2:0]</th>
<th>OC5PE</th>
<th>OC5FE</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable
Refer to OC1CE description.
Bits 24, 14, 13, 12 OC6M[3:0]: Output compare 6 mode
   Refer to OC1M description.
   Bit 11 OC6PE: Output compare 6 preload enable
   Refer to OC1PE description.
   Bit 10 OC6FE: Output compare 6 fast enable
   Refer to OC1FE description.
   Bits 9:8 Reserved, must be kept at reset value.
   Bit 7 OC5CE: Output compare 5 clear enable
   Refer to OC1CE description.
   Bits 16, 6, 5, 4 OC5M[3:0]: Output compare 5 mode
   Refer to OC1M description.
   Bit 3 OC5PE: Output compare 5 preload enable
   Refer to OC1PE description.
   Bit 2 OC5FE: Output compare 5 fast enable
   Refer to OC1FE description.
   Bits 1:0 Reserved, must be kept at reset value.

20.4.25 TIM1 capture/compare register 5 (TIM1_CCR5)

Address offset: 0x58
Reset value: 0x0000 0000
Bit 31 **GC5C3**: Group Channel 5 and Channel 3  
Distortion on Channel 3 output:  
0: No effect of OC5REF on OC3REFC  
1: OC3REFC is the logical AND of OC3REFC and OC5REF  
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMx_CCMR2).  
*Note: it is also possible to apply this distortion on combined PWM signals.*

Bit 30 **GC5C2**: Group Channel 5 and Channel 2  
Distortion on Channel 2 output:  
0: No effect of OC5REF on OC2REFC  
1: OC2REFC is the logical AND of OC2REFC and OC5REF  
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMx_CCMR1).  
*Note: it is also possible to apply this distortion on combined PWM signals.*

Bit 29 **GC5C1**: Group Channel 5 and Channel 1  
Distortion on Channel 1 output:  
0: No effect of OC5REF on OC1REFC  
1: OC1REFC is the logical AND of OC1REFC and OC5REF  
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMx_CCMR1).  
*Note: it is also possible to apply this distortion on combined PWM signals.*

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR5[15:0]**: Capture/Compare 5 value  
CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).  
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.  
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC5 output.

### 20.4.26 TIM1 capture/compare register 6 (TIM1_CCR6)

Address offset: 0x5C  
Reset value: 0x0000

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

Bits 15:0 **CCR6[15:0]**: Capture/Compare 6 value  
CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).  
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.  
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC6 output.
### 20.4.27 TIM1 alternate function option register 1 (TIM1_AF1)

Address offset: 0x60  
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bit 31:18</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 17:14</td>
<td><strong>ETRSEL[3:0]</strong>: ETR source selection</td>
</tr>
<tr>
<td></td>
<td>These bits select the ETR input source.</td>
</tr>
<tr>
<td></td>
<td>0000: ETR legacy mode</td>
</tr>
<tr>
<td></td>
<td>0001: COMP1 output</td>
</tr>
<tr>
<td></td>
<td>0010: COMP2 output</td>
</tr>
<tr>
<td></td>
<td>0011: ADC1 AWD1</td>
</tr>
<tr>
<td></td>
<td>0100: ADC1 AWD2</td>
</tr>
<tr>
<td></td>
<td>0101: ADC1 AWD3</td>
</tr>
<tr>
<td></td>
<td>Others: Reserved</td>
</tr>
<tr>
<td></td>
<td>Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</td>
</tr>
<tr>
<td>Bit 13:12</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 11</td>
<td><strong>BKCMP2P</strong>: BRK COMP2 input polarity</td>
</tr>
<tr>
<td></td>
<td>This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.</td>
</tr>
<tr>
<td></td>
<td>0: COMP2 input is active high</td>
</tr>
<tr>
<td></td>
<td>1: COMP2 input is active low</td>
</tr>
<tr>
<td></td>
<td>Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</td>
</tr>
<tr>
<td>Bit 10</td>
<td><strong>BKCMP1P</strong>: BRK COMP1 input polarity</td>
</tr>
<tr>
<td></td>
<td>This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.</td>
</tr>
<tr>
<td></td>
<td>0: COMP1 input is active high</td>
</tr>
<tr>
<td></td>
<td>1: COMP1 input is active low</td>
</tr>
<tr>
<td></td>
<td>Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</td>
</tr>
<tr>
<td>Bit 9</td>
<td><strong>BKINP</strong>: BRK BKIN input polarity</td>
</tr>
<tr>
<td></td>
<td>This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.</td>
</tr>
<tr>
<td></td>
<td>0: BKIN input is active high</td>
</tr>
<tr>
<td></td>
<td>1: BKIN input is active low</td>
</tr>
<tr>
<td></td>
<td>Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</td>
</tr>
<tr>
<td>Bits 8:3</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
20.4.28 TIM1 Alternate function register 2 (TIM1_AF2)

Address offset: 0x64

Reset value: 0x0000 0001

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BK2CMP2P**: BRK2 COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: COMP2 input is active low
1: COMP2 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 10  **BK2CMP1P**: BRK2 COMP1 input polarity
This bit selects the COMP1 input sensitivity. It must be programmed together with the BK2P polarity bit.
0: COMP1 input is active low
1: COMP1 input is active high
*Note:* *This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 9  **BK2INP**: BRK2 BKIN2 input polarity
This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.
0: BKIN2 input is active low
1: BKIN2 input is active high
*Note:* *This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bits 8:3  Reserved, must be kept at reset value.

Bit 2  **BK2CMP2E**: BRK2 COMP2 enable
This bit enables the COMP2 for the timer’s BRK2 input. COMP2 output is ‘ORed’ with the other BRK2 sources.
0: COMP2 input disabled
1: COMP2 input enabled
*Note:* *This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 1  **BK2CMP1E**: BRK2 COMP1 enable
This bit enables the COMP1 for the timer’s BRK2 input. COMP1 output is ‘ORed’ with the other BRK2 sources.
0: COMP1 input disabled
1: COMP1 input enabled
*Note:* *This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 0  **BK2INE**: BRK2 BKIN input enable
This bit enables the BKIN2 alternate function input for the timer’s BRK2 input. BKIN2 input is ‘ORed’ with the other BRK2 sources.
0: BKIN2 input disabled
1: BKIN2 input enabled
*Note:* *This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

*Note:* Refer to *Figure 146: Break and Break2 circuitry overview.*
20.4.29 TIM1 timer input selection register (TIM1_TISEL)

Address offset: 0x68
Reset value: 0x0000 0000

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

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

Bits 27:24 **TI4SEL[3:0]** selects TI4[0] to TI4[15] input
  0000: TIM1_CH4 input
  Others: Reserved

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

Bits 19:16 **TI3SEL[3:0]** selects TI3[0] to TI3[15] input
  0000: TIM1_CH3 input
  Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]** selects TI2[0] to TI2[15] input
  0000: TIM1_CH2 input
  0001: COMP2 output
  Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]** selects TI1[0] to TI1[15] input
  0000: TIM1_CH1 input
  0001: COMP1 output
  Others: Reserved
### 20.4.30 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

| Offset | Register name   | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | TIM1_CR1       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | TIM1_CR2       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x08   | TIM1_SMCR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0C   | TIM1_DIER      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | TIM1_SR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x14   | TIM1_EGR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| Offset | Register name   | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x18   | TIM1_CCMR1     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Output Compare mode |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1C   | TIM1_CCMR2     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Output Compare mode |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x20   | TIM1_CCER      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
### Table 107. TIM1 register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x24</td>
<td>TIM1_CNT</td>
<td>CNT[15:0]</td>
<td>0x28</td>
<td>TIM1_PSC</td>
<td>PSC[15:0]</td>
<td>0x2C</td>
<td>TIM1_ARR</td>
<td>ARR[15:0]</td>
<td>0x30</td>
<td>TIM1_RCR</td>
<td>REP[15:0]</td>
<td>0x34</td>
<td>TIM1_CCR1</td>
<td>CCR1[15:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td></td>
<td>Reset value</td>
<td>1</td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td></td>
<td>Reset value</td>
<td>0</td>
</tr>
<tr>
<td>0x38</td>
<td>TIM1_CCR2</td>
<td>CCR2[15:0]</td>
<td>0x3C</td>
<td>TIM1_CCR3</td>
<td>CCR3[15:0]</td>
<td>0x40</td>
<td>TIM1_CCR4</td>
<td>CCR4[15:0]</td>
<td>0x44</td>
<td>TIM1_BDTR</td>
<td>LOC[1:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td></td>
<td>Reset value</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x48</td>
<td>TIM1_DCR</td>
<td>DBL[4:0]</td>
<td></td>
<td>DMAR</td>
<td>DMAB[31:0]</td>
<td>0x50</td>
<td>TIM1_OR1</td>
<td></td>
<td>0x54</td>
<td>TIM1_CCMR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Output</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 107 Notes:**
- **TIM1_CNT**: CNT[15:0] reset value: 0
- **TIM1_PSC**: PSC[15:0] reset value: 0
- **TIM1_ARR**: ARR[15:0] reset value: 1
- **TIM1_RCR**: REP[15:0] reset value: 0
- **TIM1_CCR1**: CCR1[15:0] reset value: 0
- **TIM1_CCR2**: CCR2[15:0] reset value: 0
- **TIM1_CCR3**: CCR3[15:0] reset value: 0
- **TIM1_CCR4**: CCR4[15:0] reset value: 0
- **TIM1_BDTR**: LOC[1:0] reset value: 0
- **DMAR**: DMAB[31:0] reset value: 0
- **TIM1_OR1**: reset value: 0
- **TIM1_CCMR3**: Output Compare mode reset value: 0

**Register Map:**
- TIM1_CNT: CNT[15:0]
- TIM1_PSC: PSC[15:0]
- TIM1_ARR: ARR[15:0]
- TIM1_RCR: REP[15:0]
- TIM1_CCR1: CCR1[15:0]
- TIM1_CCR2: CCR2[15:0]
- TIM1_CCR3: CCR3[15:0]
- TIM1_CCR4: CCR4[15:0]
- TIM1_BDTR: LOC[1:0]
- DMAR: DMAB[31:0]
- TIM1.OR1: reset value: 0
- TIM1.CCMR3: Output Compare mode reset value: 0
Table 107. TIM1 register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x58</td>
<td>TIM1_CCR5</td>
<td>0x5C</td>
<td>TIM1_CCR6</td>
<td>0x60</td>
<td>TIM1_AF1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x64</td>
<td>TIM1_AF2</td>
<td>0x68</td>
<td>TIM1_TISEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>CCR5[15:0]</td>
<td>Reset value</td>
<td>CCR6[15:0]</td>
<td>Reset value</td>
<td>ETRSEL [3:0]</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>Offset</td>
<td>Register name</td>
<td>Offset</td>
<td>Register name</td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
<td>--------</td>
<td>--------------</td>
<td>--------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.2 on page 54 for the register boundary addresses.
21 General-purpose timers (TIM2/TIM3)

21.1 TIM2/TIM3 introduction

The general-purpose timers consist of a 16-bit/32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in Section 21.3.19: Timer synchronization.

21.2 TIM2/TIM3 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3) or 32-bit (TIM2) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management
Figure 163. General-purpose timer block diagram
21.3 **TIM2/TIM3 functional description**

21.3.1 **Time-base unit**

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

**Prescaler description**

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 164* and *Figure 165* give some examples of the counter behavior when the prescaler ratio is changed on the fly:
Figure 164. Counter timing diagram with prescaler division change from 1 to 2

Figure 165. Counter timing diagram with prescaler division change from 1 to 4
21.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

**Figure 166. Counter timing diagram, internal clock divided by 1**
Figure 167. Counter timing diagram, internal clock divided by 2

Figure 168. Counter timing diagram, internal clock divided by 4
Figure 169. Counter timing diagram, internal clock divided by N

Figure 170. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)
Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An Update event can be generate at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn’t change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.
The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

**Figure 172. Counter timing diagram, internal clock divided by 1**

**Figure 173. Counter timing diagram, internal clock divided by 2**
Figure 174. Counter timing diagram, internal clock divided by 4

CK_PSC

CNT_EN

Timer clock = CK_CNT

Counter register

0001 0000 0000 0001

Counter underflow

Update event (UEV)

Update interrupt flag (UIF)

Figure 175. Counter timing diagram, internal clock divided by N

CK_PSC

Timer clock = CK_CNT

Counter register

20 1F 00 36

Counter underflow

Update event (UEV)

Update interrupt flag (UIF)
Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to ‘00’. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 177. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6

1. Here, center-aligned mode 1 is used (for more details refer to Section 21.4.1: TIMx control register 1 (TIMx_CR1)\(x = 2 \text{ to } 3\) on page 630).
1. Center-aligned mode 2 or 3 is used with an UIF on overflow.
**Figure 180. Counter timing diagram, internal clock divided by N**

- **CK_PSC**
- **Timer clock = CK_CNT**
- **Counter register**: 20 1F 01 00
- **Counter underflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**

**Figure 181. Counter timing diagram, Update event with ARPE=1 (counter underflow)**

- **CK_PSC**
- **CEN**
- **Timer clock = CK_CNT**
- **Counter register**: 05 04 03 02 01 00 01 02 03 04 05 06 07
- **Counter underflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**
- **Auto-reload preload register**: FD 36
- **Write a new value in TIMx_ARR**
- **Auto-reload active register**: FD 36
21.3.3 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin (TiX)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, Timer X can be configured to act as a prescaler for Timer Y. Refer to : Using one timer as prescaler for another timer on page 625 for more details.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 183 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.
External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

1. Codes ranging from 01000 to 11111: ITRy.

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S=’01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.
The delay between the rising edge on TI2 and the actual clock of the counter is due to the
resynchronization circuit on TI2 input.

Figure 185. Control circuit in external clock mode 1

External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.
The counter can count at each rising or falling edge on the external trigger input ETR.
Figure 186 gives an overview of the external trigger input block.
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. Select the proper ETR source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
2. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register.
4. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register.
5. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETPS prescaler setting.
21.3.4 Capture/Compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.
The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.
21.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.

2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.

3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

4. Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).

5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).

6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.

7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.
21.3.6 PWM input mode

This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to ’0’ and the CC1NP bit to ’0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to ’1’ and the CC2NP bit to ’0’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ’1’ in the TIMx_CCER register.

Figure 191. PWM input mode timing

1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.
21.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (ocxref/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

21.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, one must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.
The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 192.

**Figure 192. Output compare mode, toggle on OC1**

The PWM mode

Pulse width modulation mode allows to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). However, to comply with the OCREF_CLR functionality (OCREF can be...
cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison or
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=’000) to one of the PWM modes (OCxM=’110 or ’111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

**PWM edge-aligned mode**

**Upcounting configuration**

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to [Upcounting mode](#) on page 590.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1. If the compare value is 0 then OCxREF is held at ‘0. Figure 193 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

**Figure 193. Edge-aligned PWM waveforms (ARR=8)**
**Downcounting configuration**

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to *Downcounting mode on page 593*.

In PWM mode 1, the reference signal ocxref is low as long as TIMx_CNT> TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then ocxref is held at 100%. PWM is not possible in this mode.

**PWM center-aligned mode**

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting) on page 596*.

*Figure 194* shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.
Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.

- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.
21.3.10 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:
- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 2.

Figure 195 shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1).

![Figure 195. Generation of 2 phase-shifted PWM signals with 50% duty cycle](image)

21.3.11 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:
- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

**Note:** The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

*Figure 196* shows an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:
- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1

**Figure 196. Combined PWM mode on channels 1 and 3**

---

### 21.3.12 Clearing the OCxREF signal on an external event

The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

OCREF_CLR_INPUT can be selected between the OCREF_CLR input and ETRF (ETR after the filter) by configuring the OCCS bit in the TIMx_SMCR register.
The OCxREF signal for a given channel can be reset by applying a high level on the ETRF input (OCxCE enable bit set to 1 in the corresponding TIMx_CCMRx register). OCxREF remains low until the next update event (UEV) occurs.

This function can be used only in the output compare and PWM modes. It does not work in forced mode.

For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 197 shows the behavior of the OCxREF signal when the ETRF input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

**Figure 197. Clearing TIMx OCxREF**

Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.
### 21.3.13 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- $\text{CNT} < \text{CCR}_x \leq \text{ARR}$ (in particular, $0 < \text{CCR}_x$),

**Figure 198. Example of one-pulse mode.**

![Diagram](image)

For example one may want to generate a positive pulse on OC1 with a length of $t_{\text{PULSE}}$ and after a delay of $t_{\text{DELAY}}$ as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 on TI2 by writing CC2S=01 in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110 in the TIMx_SMCR register (trigger mode).
The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The \( t_{\text{DELAY}} \) is defined by the value written in the TIMx_CCR1 register.
- The \( t_{\text{PULSE}} \) is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one wants to build a waveform with a transition from '0 to '1 when a compare match occurs and a transition from '1 to '0 when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0 in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low. Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

**Particular case: OCx fast enable:**

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay \( t_{\text{DELAY min}} \) we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 21.3.14 Retriggerable one pulse mode (OPM)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in Section 21.3.13:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

**Note:** In retriggerable one pulse mode, the CCxIF flag is not significant.

*The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.*

*This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.*
21.3.15 Encoder interface mode

To select Encoder Interface mode write SMS='001 in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to Table 108. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to ‘1’). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder’s position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.
An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder’s differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

**Figure 200** gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= 01 (TIMx_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P and CC1NP = ‘0’ (TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P and CC2NP = ‘0’ (TIMx_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx_CR1 register, Counter is enabled)

**Figure 200. Example of counter operation in encoder interface mode**

<table>
<thead>
<tr>
<th>Active edge</th>
<th>Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)</th>
<th>TI1FP1 signal</th>
<th>TI2FP2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>TI1 only</td>
<td>High</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>Down</td>
</tr>
<tr>
<td>TI2 only</td>
<td>High</td>
<td>No Count</td>
<td>No Count</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>No Count</td>
<td>No Count</td>
</tr>
<tr>
<td>TI1 and TI2</td>
<td>High</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>Down</td>
</tr>
</tbody>
</table>

| Counting on | Rising   | Falling  | Rising | Falling |
| T1 only     |          |          |        |         |
| T2 only     |          |          |        |         |
| T1 and T2   |          |          |        |         |

**Table 108. Counting direction versus encoder signals**

**Figure 201** gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).
The timer, when configured in Encoder Interface mode provides information on the sensor’s current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

21.3.16 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register’s bit 31 (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter’s most significant bit is only accessible in write mode).

21.3.17 Timer input XOR function

The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1 to TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in Section 20.3.25: Interfacing with Hall sensors on page 537.
21.3.18 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).

2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

![Figure 202. Control circuit in reset mode](MS31401V2)

Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:
1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCRER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

**Slave mode: Trigger mode**

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write...
CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

**Figure 204. Control circuit in trigger mode**

**Slave mode: External Clock mode 2 + trigger mode**

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.

2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01 in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edge only).

3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

**Figure 205. Control circuit in external clock mode 2 + trigger mode**

21.3.19 **Timer synchronization**

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

*Figure 206: Master/Slave timer example* presents an overview of the trigger selection and the master mode selection blocks.
Using one timer as prescaler for another timer

For example, TIM3 can be configured to act as a prescaler for TIM2. Refer to Figure 206. To do this:

1. Configure TIM3 in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM3_CR2 register, a rising edge is output on TRGO each time an update event is generated.

2. To connect the TRGO output of TIM3 to TIM2, TIM2 must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM2_SMCR register (writing TS=00010).

3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM3 trigger signal (which correspond to the TIM3 counter overflow).

4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If OCx is selected on TIM3 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 3. Refer to Figure 206 for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM3 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT \( (f_{CK\_CNT} = f_{CK\_INT}/3) \).

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).

2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).

3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).

4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).

5. Enable TIM2 by writing '1 in the CEN bit (TIM2_CR1 register).

6. Start TIM3 by writing '1 in the CEN bit (TIM3_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.

Figure 207. Gating TIM2 with OC1REF of TIM3
In the example in Figure 207, the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM3. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to Figure 208), we synchronize TIM3 and TIM2. TIM3 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM3 is disabled by writing '0 to the CEN bit in the TIM3_CR1 register:

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Reset TIM3 by writing '1 in UG bit (TIM3_EGR register).
6. Reset TIM2 by writing '1 in UG bit (TIM2_EGR register).
7. Initialize TIM2 to 0xE7 by writing '0xE7' in the TIM2 counter (TIM2_CNTL).
8. Enable TIM2 by writing '1 in the CEN bit (TIM2_CR1 register).
9. Start TIM3 by writing '1 in the CEN bit (TIM3_CR1 register).
10. Stop TIM3 by writing '0 in the CEN bit (TIM3_CR1 register).

**Figure 208. Gating TIM2 with Enable of TIM3**

Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 3. Refer to Figure 206 for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write '0 to the CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT (f_{CK_CNT} = f_{CK_INT}/3).
1. Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3_CR2 register).
2. Configure the TIM3 period (TIM3_ARR registers).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
5. Start TIM3 by writing ‘1’ in the CEN bit (TIM3_CR1 register).

Figure 209. Triggering TIM2 with update of TIM3

As in the previous example, both counters can be initialized before starting counting. Figure 210 shows the behavior with the same configuration as in Figure 209 but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Figure 210. Triggering TIM2 with Enable of TIM3

Starting 2 timers synchronously in response to an external trigger

In this example, we set the enable of TIM3 when its TI1 input rises, and the enable of TIM2 with the enable of TIM3. Refer to Figure 206 for connections. To ensure the counters are
General-purpose timers (TIM2/TIM3) RM0444

aligned, TIM3 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIM2):

1. Configure TIM3 master mode to send its Enable as trigger output (MMS=001 in the TIM3_CR2 register).
2. Configure TIM3 slave mode to get the input trigger from TI1 (TS=00100 in the TIM3_SMCR register).
3. Configure TIM3 in trigger mode (SMS=110 in the TIM3_SMCR register).
4. Configure the TIM3 in Master/Slave mode by writing MSM=1 (TIM3_SMCR register).
5. Configure TIM2 to get the input trigger from TIM3 (TS=00000 in the TIM2_SMCR register).
6. Configure TIM2 in trigger mode (SMS=110 in the TIM2_SMCR register).

When a rising edge occurs on TI1 (TIM3), both counters starts counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on TIM3.

Figure 211. Triggering TIM3 and TIM2 with TIM3 TI1 input

<table>
<thead>
<tr>
<th>CK_INT</th>
<th>TIM3-TI1</th>
<th>TIM3-CEN=CNT_EN</th>
<th>TIM3-CK_PSC</th>
<th>TIM3-CNT</th>
<th>TIM3-TIF</th>
<th>TIM2-CEN=CNT_EN</th>
<th>TIM2-CK_PSC</th>
<th>TIM2-CNT</th>
<th>TIM2-TIF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

21.3.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.
The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address, i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let’s take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

### 21.3.21 Debug mode

When the microcontroller enters debug mode (Cortex®-M0+ core - halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBGMCU module. For more details, refer to Section 37.9.2: Debug support for timers, watchdog and I²C.
21.4 TIM2/TIM3 registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

21.4.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 3)

Address offset: 0x00

Reset value: 0x0000

| Bit 15:12 | Reserved, must be kept at reset value. |
| Bit 11   | UIFREMAP: UIF status bit remapping |
| 0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31. |
| 1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31. |
| Bit 10   | Reserved, must be kept at reset value. |
| Bits 9:8 | CKD[1:0]: Clock division |
| 00: tDTS = tCK_INT |
| 01: tDTS = 2 × tCK_INT |
| 10: tDTS = 4 × tCK_INT |
| 11: Reserved |
| Bit 7    | ARPE: Auto-reload preload enable |
| 0: TIMx_ARR register is not buffered |
| 1: TIMx_ARR register is buffered |
| Bits 6:5 | CMS[1:0]: Center-aligned mode selection |
| 00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR). |
| 01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up. |
| 10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up. |
| 11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down. |
| Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1) |
| Bit 4    | DIR: Direction |
| 0: Counter used as upcounter |
| 1: Counter used as downcounter |
| Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode. |
Bit 3  **OPM:** One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2  **URS:** Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled.
These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1  **UDIS:** Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value
(ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is
set or if a hardware reset is received from the slave mode controller.

Bit 0  **CEN:** Counter enable
0: Counter disabled
1: Counter enabled

*Note: External clock, gated mode and encoder mode can work only if the CEN bit has been
previously set by software. However trigger mode can set the CEN bit automatically by
hardware.*

CEN is cleared automatically in one-pulse mode, when an update event occurs.

### 21.4.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 3)

Address offset: 0x04
Reset value: 0x0000

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

TI1S | MMS[2:0] | CCDS | .... | .... | .... | .... | .... | .... | .... | .... | .... | .... | .... | .... | .... |

rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
Bits 15:8  Reserved, must be kept at reset value.

Bit 7  **TI1S**: TI1 selection

- 0: The TIMx_CH1 pin is connected to TI1 input
- 1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

See also [Section 20.3.25: Interfacing with Hall sensors on page 537](#)

Bits 6:4  **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

- 000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

- 001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.

- 010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

- 011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred.

- 100: **Compare** - OC1REF signal is used as trigger output (TRGO)
- 101: **Compare** - OC2REF signal is used as trigger output (TRGO)
- 110: **Compare** - OC3REF signal is used as trigger output (TRGO)
- 111: **Compare** - OC4REF signal is used as trigger output (TRGO)

**Note**: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3  **CCDS**: Capture/compare DMA selection

- 0: CCx DMA request sent when CCx event occurs
- 1: CCx DMA requests sent when update event occurs

Bits 2:0  Reserved, must be kept at reset value.
21.4.3  **TIMx slave mode control register (TIMx_SMCR)(x = 2 to 3)**

Address offset: 0x08

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:22</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>19:17</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td><strong>SMS[3]</strong>: Slave mode selection - bit 3</td>
<td>Refer to SMS description - bits 2:0.</td>
</tr>
</tbody>
</table>
| 15   | **ETP**: External trigger polarity | This bit selects whether ETR or ETR is used for trigger operations  
|      |                                    | 0: ETR is non-inverted, active at high level or rising edge  
|      |                                    | 1: ETR is inverted, active at low level or falling edge |
| 14   | **ECE**: External clock enable    | This bit enables External clock mode 2.  
|      |                                    | 0: External clock mode 2 disabled  
|      |                                    | 1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.  
|      |                                    | 1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).  
|      |                                    | 2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).  
|      |                                    | 3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF. |
| 13:12 | **ETPS[1:0]**: External trigger prescaler | External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.  
|      |                                    | 00: Prescaler OFF  
|      |                                    | 01: ETRP frequency divided by 2  
|      |                                    | 10: ETRP frequency divided by 4  
|      |                                    | 11: ETRP frequency divided by 8 |

Bits 31:22  Reserved, must be kept at reset value.

Bits 19:17  Reserved, must be kept at reset value.

Bit 16  **SMS[3]**: Slave mode selection - bit 3  
Refer to SMS description - bits 2:0.

Bit 15  **ETP**: External trigger polarity  
This bit selects whether ETR or ETR is used for trigger operations  
0: ETR is non-inverted, active at high level or rising edge  
1: ETR is inverted, active at low level or falling edge

Bit 14  **ECE**: External clock enable  
This bit enables External clock mode 2.  
0: External clock mode 2 disabled  
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.  
1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).  
2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).  
3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12  **ETPS[1:0]**: External trigger prescaler  
External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.  
00: Prescaler OFF  
01: ETRP frequency divided by 2  
10: ETRP frequency divided by 4  
11: ETRP frequency divided by 8
Bits 11:8 **ETF[3:0]:** External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at fDTS
0001: fSAMPLING= fCK_INT, N=2
0010: fSAMPLING= fCK_INT, N=4
0011: fSAMPLING= fCK_INT, N=8
0100: fSAMPLING= fDTS/2, N=6
0101: fSAMPLING= fDTS/2, N=8
0110: fSAMPLING= fDTS/4, N=6
0111: fSAMPLING= fDTS/4, N=8
1000: fSAMPLING= fDTS/8, N=6
1001: fSAMPLING= fDTS/8, N=8
1010: fSAMPLING= fDTS/16, N=5
1011: fSAMPLING= fDTS/16, N=6
1100: fSAMPLING= fDTS/16, N=8
1101: fSAMPLING= fDTS/32, N=5
1110: fSAMPLING= fDTS/32, N=6
1111: fSAMPLING= fDTS/32, N=8

Bit 7 **MSM:** Master/Slave mode

0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.
Bits 21, 20, 6, 5, 4  **TS[4:0]:** Trigger selection (see bits 21:20 for TS[4:3])

This bit-field selects the trigger input to be used to synchronize the counter.

- 00000: Internal Trigger 0 (ITR0)
- 00001: Internal Trigger 1 (ITR1)
- 00010: Internal Trigger 2 (ITR2)
- 00011: Internal Trigger 3 (ITR3)
- 00100: TI1 Edge Detector (TI1F_ED)
- 00101: Filtered Timer Input 1 (TI1FP1)
- 00110: Filtered Timer Input 2 (TI2FP2)
- 00111: External Trigger input (ETRF)
- 01000: Internal Trigger 4 (ITR4)
- 01001: Internal Trigger 5 (ITR5)
- 01010: Internal Trigger 6 (ITR6)
- 01011: Internal Trigger 7 (ITR7)
- 01100: Internal Trigger 8 (ITR8)

Others: Reserved

See Table 109: TIMx internal trigger connection on page 636 for more details on ITRx meaning for each Timer.

**Note:** These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3  **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source.

- 0: OCREF_CLR_INT is connected to COMP1 or COMP2 output depending on TIMx.OR1.OCREF_CLR
- 1: OCREF_CLR_INT is connected to ETRF
Bits 16, 2, 1, 0  **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ’1 then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

**Note:** The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=00100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

**Note:** The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

<table>
<thead>
<tr>
<th>Slave TIM</th>
<th>ITR0</th>
<th>ITR1</th>
<th>ITR2</th>
<th>ITR3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM2</td>
<td>TIM1</td>
<td>TIM15</td>
<td>TIM3</td>
<td>TIM14_OC1</td>
</tr>
<tr>
<td>TIM3</td>
<td>TIM1</td>
<td>TIM2</td>
<td>TIM15</td>
<td>TIM14_OC1</td>
</tr>
</tbody>
</table>

**21.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 3)**

Address offset: 0x0C

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Res</td>
<td>TDE</td>
<td>Res</td>
<td>CC4DE</td>
<td>CC3DE</td>
<td>CC2DE</td>
<td>CC1DE</td>
<td>UDE</td>
<td>Res</td>
<td>TIE</td>
<td>Res</td>
<td>CC4IE</td>
<td>CC3IE</td>
<td>CC2IE</td>
<td>CC1IE</td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bit 15**  Reserved, must be kept at reset value.

**Bit 14**  **TDE:** Trigger DMA request enable

0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

**Bit 13**  Reserved, must be kept at reset value.
Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
- 0: CC4 DMA request disabled.
- 1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
- 0: CC3 DMA request disabled.
- 1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
- 0: CC2 DMA request disabled.
- 1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
- 0: CC1 DMA request disabled.
- 1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
- 0: Update DMA request disabled.
- 1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
- 0: Trigger interrupt disabled.
- 1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
- 0: CC4 interrupt disabled.
- 1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
- 0: CC3 interrupt disabled.
- 1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
- 0: CC2 interrupt disabled.
- 1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
- 0: CC1 interrupt disabled.
- 1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
- 0: Update interrupt disabled.
- 1: Update interrupt enabled.

### 21.4.5 TIMx status register (TIMx_SR)(x = 2 to 3)

Address offset: 0x10

Reset value: 0x0000

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

- **CC4OF**: Capture/Compare 4 overflow
- **CC3OF**: Capture/Compare 3 overflow
- **CC2OF**: Capture/Compare 2 overflow
- **CC1OF**: Capture/Compare 1 overflow
- **CC4IF**: Capture/Compare 4 interrupt flag
- **CC3IF**: Capture/Compare 3 interrupt flag
- **CC2IF**: Capture/Compare 2 interrupt flag
- **CC1IF**: Capture/Compare 1 interrupt flag
- **UIF**: Update interrupt flag

rc_w0: Reserved, must be kept at reset value.
Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
Refer to CC1IF description
Bit 2  **CC2IF**: Capture/Compare 2 interrupt flag
   Refer to CC1IF description

Bit 1  **CC1IF**: Capture/compare 1 interrupt flag
   This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
   0: No compare match / No input capture occurred
   1: A compare match or an input capture occurred

**If channel CC1 is configured as output**: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

**If channel CC1 is configured as input**: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0  **UIF**: Update interrupt flag
   This bit is set by hardware on an update event. It is cleared by software.
   0: No update occurred
   1: Update interrupt pending. This bit is set by hardware when the registers are updated:
      At overflow or underflow and if UDIS=0 in the TIMx_CR1 register.
      When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
      When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

### 21.4.6 TIMx event generation register (TIMx_EGR)(x = 2 to 3)

Address offset: 0x14

Reset value: 0x0000

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |

Bits 15:7 Reserved, must be kept at reset value.

Bit 6  **TG**: Trigger generation
   This bit is set by software in order to generate an event, it is automatically cleared by hardware.
   0: No action
   1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4  **CC4G**: Capture/compare 4 generation
   Refer to CC1G description

Bit 3  **CC3G**: Capture/compare 3 generation
   Refer to CC1G description
21.4.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 3)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Reset Value</th>
<th>Access</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
<td></td>
<td>rw</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td>rw</td>
<td>Bits 15:12 <strong>IC2F[3:0]</strong>: Input capture 2 filter</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td>rw</td>
<td>Bits 11:10 <strong>IC2PSC[1:0]</strong>: Input capture 2 prescaler</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to **CC1G** description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:

**If channel CC1 is configured as output:**
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

**If channel CC1 is configured as input:**
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).
Bits 9:8  **CC2S[1:0]:** Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output.
01: CC2 channel is configured as input, IC2 is mapped on TI2.
10: CC2 channel is configured as input, IC2 is mapped on TI1.
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register).

*Note:* CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4  **IC1F[3:0]:** Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:
0000: No filter, sampling is done at $f_{\text{DTS}}$
0001: $f_{\text{SAMPLING}} = f_{\text{CK INT}}$, $N=2$
0010: $f_{\text{SAMPLING}} = f_{\text{CK INT}}$, $N=4$
0011: $f_{\text{SAMPLING}} = f_{\text{CK INT}}$, $N=8$
0100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, $N=6$
0101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, $N=8$
0110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, $N=6$
0111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, $N=8$
1000: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, $N=6$
1001: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, $N=8$
1010: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, $N=5$
1011: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, $N=6$
1100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N=5$
1101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N=6$
1110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N=8$
1111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N=8$

Bits 3:2  **IC1PSC[1:0]:** Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E=0 (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0  **CC1S[1:0]:** Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register).

*Note:* CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).
### 21.4.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)

(x = 2 to 3)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

#### Output compare mode:

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

Bits 31:25: Reserved, must be kept at reset value.

Bits 23:17: Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14: 12 **OC2M[3:0]**: Output compare 2 mode

- refer to **OC1M** description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

- This bit-field defines the direction of the channel (input/output) as well as the used input.
  - 00: CC2 channel is configured as output
  - 01: CC2 channel is configured as input, IC2 is mapped on TI2
  - 10: CC2 channel is configured as input, IC2 is mapped on TI1
  - 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (**CC2E** = 0 in **TIMx_CCMR1**).

Bit 7 **OC1CE**: Output compare 1 clear enable

- 0: OC1Ref is not affected by the ETRF input
- 1: OC1Ref is cleared as soon as a High level is detected on ETRF input
Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. (this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0') as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes active again at the next update.

1010: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1011: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1100: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1101: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

**Note:** In PWM mode, the OCRE level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.
Bit 3 **OC1PE**: Output compare 1 preload enable
- 0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
- 1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

**Note**: The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output compare 1 fast enable
This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
- 0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
- 1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
- 00: CC1 channel is configured as output.
- 01: CC1 channel is configured as input, IC1 is mapped on TI1.
- 10: CC1 channel is configured as input, IC1 is mapped on TI2.
- 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

**Note**: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

21.4.9 **TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)**
(x = 2 to 3)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Input capture mode**:

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler
21.4.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)  
(x = 2 to 3)  

Address offset: 0x1C  
Reset value: 0x0000 0000  

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.  

Output compare mode:

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;OC4M [3]&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;OC4CE&quot;</td>
<td>&quot;OC4M[2:0]&quot;</td>
<td>&quot;OC4PE&quot;</td>
<td>&quot;OC4FE&quot;</td>
<td>&quot;CC4S[1:0]&quot;</td>
<td>&quot;OC3CE&quot;</td>
<td>&quot;OC3M[2:0]&quot;</td>
<td>&quot;OC3PE&quot;</td>
<td>&quot;OC3FE&quot;</td>
<td>&quot;CC3S[1:0]&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.  
Bits 23:17 Reserved, must be kept at reset value.  
Bit 15 OC4CE: Output compare 4 clear enable  

Bits 24, 14:12 OC4M[3:0]: Output compare 4 mode  
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)  

Bit 11 OC4PE: Output compare 4 preload enable  
Bit 10 OC4FE: Output compare 4 fast enable
General-purpose timers (TIM2/TIM3)  

**21.4.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 3)**

Address offset: 0x20  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC4NP</td>
<td>CC4P</td>
<td>CC4E</td>
<td>CC3NP</td>
<td>CC3P</td>
<td>CC3E</td>
<td>CC2NP</td>
<td>CC2P</td>
<td>CC2E</td>
<td>CC1NP</td>
<td>CC1P</td>
<td>CC1E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bit 15 **CC4NP**: Capture/Compare 4 output Polarity.  
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output Polarity.  
Refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable.  
Refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 output Polarity.  
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: Capture/Compare 3 output Polarity.  
Refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable.  
Refer to CC1E description
Bit 7  **CC2NP**: Capture/Compare 2 output Polarity.
   Refer to CC1NP description

Bit 6  Reserved, must be kept at reset value.

Bit 5  **CC2P**: Capture/Compare 2 output Polarity.
   refer to CC1P description

Bit 4  **CC2E**: Capture/Compare 2 output enable.
   Refer to CC1E description

Bit 3  **CC1NP**: Capture/Compare 1 output Polarity.
   **CC1 channel configured as output**: CC1NP must be kept cleared in this case.
   **CC1 channel configured as input**: This bit is used in conjunction with CC1P to define
   TI1FP1/TI2FP1 polarity. refer to CC1P description.

Bit 2  Reserved, must be kept at reset value.

Bit 1  **CC1P**: Capture/Compare 1 output Polarity.
   0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
   1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
   **When CC1 channel is configured as input**, both CC1NP/CC1P bits select the active
   polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
   CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge
   (capture or trigger operations in reset, external clock or trigger mode).
   TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).
   CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge
   (capture or trigger operations in reset, external clock or trigger mode).
   TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
   CC1NP=1, CC1P=1: non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising
   and falling edges (capture or trigger operations in reset, external clock
   or trigger mode). TIxFP1 is not inverted (trigger operation in gated
   mode). This configuration must not be used in encoder mode.
   CC1NP=1, CC1P=0: This configuration is reserved, it must not be used.

Bit 0  **CC1E**: Capture/Compare 1 output enable.
   0: Capture mode disabled / OC1 is not active
   1: Capture mode enabled / OC1 signal is output on the corresponding output pin

<table>
<thead>
<tr>
<th>CCxE bit</th>
<th>OCx output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output Disabled (OCx=0, OCx_EN=0)</td>
</tr>
<tr>
<td>1</td>
<td>OCx=OCxREF + Polarity, OCx_EN=1</td>
</tr>
</tbody>
</table>

**Table 110. Output control bit for standard OCx channels**

Note:  The state of the external IO pins connected to the standard OCx channels depends on the
OCx channel state and the GPIO and AFIO registers.

**21.4.12 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 3)**

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in
TIMx_CR1 register:
- This section is for UIFREMAP = 0
- Next section is for UIFREMAP = 1
General-purpose timers (TIM2/TIM3)

Address offset: 0x24
Reset value: 0x0000 0000

21.4.13 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 3)

Bit 31 CNT[31]: Most significant bit of counter value (on TIM2)
Reserved on other timers.

Bits 30:16 CNT[30:16]: Most significant part counter value (on TIM2)

Bits 15:0 CNT[15:0]: Least significant part of counter value

21.4.14 TIMx prescaler (TIMx_PSC)(x = 2 to 3)

Address offset: 0x28
Reset value: 0x0000
21.4.15 TIMx auto-reload register (TIMx_ARR)(x = 2 to 3)

Address offset: 0x2C
Reset value: 0xFFFF FFFF

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

Bits 15:0 PSC[15:0]: Prescaler value
The counter clock frequency CK_CNT is equal to fCK_PSC / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

Bits 31:16 ARR[31:16]: High auto-reload value (on TIM2)

Bits 15:0 ARR[15:0]: Low Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the Section 21.3.1: Time-base unit on page 588 for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

21.4.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 3)

Address offset: 0x34
Reset value: 0x0000 0000

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

Bits 31:16 CCR1[31:16]:

Bits 15:0 CCR1[15:0]:
21.4.17 **TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 3)**

Address offset: 0x38  
Reset value: 0x0000 0000

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

Bits 31:16 **CCR2[31:16]**: High Capture/Compare 2 value (on TIM2)  
Bits 15:0 **CCR2[15:0]**: Low Capture/Compare 2 value

*If channel CC2 is configured as output:*
CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output.

*If channel CC2 is configured as input:*
CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

21.4.18 **TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 3)**

Address offset: 0x3C  
Reset value: 0x0000 0000
21.4.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 3)

Address offset: 0x40
Reset value: 0x0000 0000

Bits 31:16 **CCR3[31:16]**: High Capture/Compare 3 value (on TIM2)

Bits 15:0 **CCR3[15:0]**: Low Capture/Compare value

If channel CC3 is configured as output:
- CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
- It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
- The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.

If channel CC3 is configured as input:
- CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

Bits 31:16 **CCR4[31:16]**: High Capture/Compare 4 value (on TIM2)

Bits 15:0 **CCR4[15:0]**: Low Capture/Compare value

1. **if CC4 channel is configured as output (CC4S bits):**
   - CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).
   - It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
   - The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.

2. **if CC4 channel is configured as input (CC4S bits in TIMx_CCMR4 register):**
   - CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.
21.4.20  TIMx DMA control register (TIMx_DCR)(x = 2 to 3)

Address offset: 0x48
Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|   |   |   |   |   |   |   |   |   |   |

Bits 15:13  Reserved, must be kept at reset value.

Bits 12:8  **DBL[4:0]: DMA burst length**
This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

- 00000: 1 transfer,
- 00001: 2 transfers,
- 00010: 3 transfers,
- ...
- 10001: 18 transfers.

Bits 7:5  Reserved, must be kept at reset value.

Bits 4:0  **DBA[4:0]: DMA base address**
This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
- 00000: TIMx_CR1
- 00001: TIMx_CR2
- 00010: TIMx_SMCR
- ...

Example: Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

21.4.21  TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 3)

Address offset: 0x4C
Reset value: 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|   |   |   |   |   |   |   |   |   |   |
| rw | rw | rw | rw | DBA[4:0] | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 15:0  **DMAB[15:0]: DMA register for burst accesses**
A read or write operation to the DMAR register accesses the register located at the address

\[(\text{TIMx}_\text{CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4\]

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).
### 21.4.22 TIM2 option register 1 (TIM2_OR1)

Address offset: 0x50  
Reset value: 0x00000

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

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

Bit 0 **IOCREF_CLR**: Ocref_clr source selection  
This bit selects the ocref_clr input source.  
0: COMP1 output is connected to the OCREF_CLR input  
1: COMP2 output is connected to the OCREF_CLR input

### 21.4.23 TIM3 option register 1 (TIM3_OR1)

Address offset: 0x50  
Reset value: 0x00000

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

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

Bit 0 **IOCREF_CLR**: Ocref_clr source selection  
This bit selects the ocref_clr input source.  
0: COMP1 output is connected to the OCREF_CLR input  
1: COMP2 output is connected to the OCREF_CLR input

### 21.4.24 TIM2 alternate function option register 1 (TIM2_AF1)

Address offset: 0x60  
Reset value: 0x00000000

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

This bit selects the ecref_clr input source.  
0: COMP1 output is connected to the OCREF_CLR input  
1: COMP2 output is connected to the OCREF_CLR input
Bits 31:18  Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]:** ETR source selection
These bits select the ETR input source.
0000: ETR legacy mode
0001: COMP1
0010: COMP2
0011: LSE
Others: Reserved

Bits 13:0  Reserved, must be kept at reset value.

### 21.4.25 TIM3 alternate function option register 1 (TIM3_AF1)

Address offset: 0x60
Reset value: 0x0000 0000

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

Bits 31:18  Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]:** ETR source selection
These bits select the ETR input source.
0000: ETR legacy mode
0001: COMP1 output
0010: COMP2 output
Others: Reserved

Bits 13:0  Reserved, must be kept at reset value.

### 21.4.26 TIM2 timer input selection register (TIM2_TISEL)

Address offset: 0x68
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TI1SEL[3:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection
These bits select the TI2[0] to TI2[15] input source.
0000: TIM2_CH2 input
0001: COMP2 output
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection
These bits select the TI1[0] to TI1[15] input source.
0000: TIM2_CH1 input
0001: COMP1 output
Others: Reserved

### 21.4.27 TIM3 timer input selection register (TIM3_TISEL)

Address offset: 0x68
Reset value: 0x0000 0000

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

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

Bits 27:24 **TI4SEL[3:0]**: TI4[0] to TI4[15] input selection
These bits select the TI4[0] to TI4[15] input source.
0000: TIM3_CH4 input
0001: COMP2 output
Others: Reserved

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

Bits 19:16 **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection
These bits select the TI3[0] to TI3[15] input source.
0000: TIM3_CH3 input
0001: COMP1 output
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.
Bits 11:8 **T12SEL[3:0]**: TI2[0] to TI2[15] input selection
These bits select the TI2[0] to TI2[15] input source.
0000: TIM3_CH2 input
0001: COMP2 output
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **T11SEL[3:0]**: TI1[0] to TI1[15] input selection
These bits select the TI1[0] to TI1[15] input source.
0000: TIM3_CH1 input
0001: COMP1 output
Others: Reserved
## 21.4.28 TIMx register map

TIMx registers are mapped as described in the table below:

<table>
<thead>
<tr>
<th>Table 111. TIM2/TIM3 register map and reset values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Offset</strong></td>
</tr>
<tr>
<td>0x00</td>
</tr>
<tr>
<td>0x04</td>
</tr>
<tr>
<td>0x08</td>
</tr>
<tr>
<td>0x14</td>
</tr>
<tr>
<td>0x1C</td>
</tr>
</tbody>
</table>
### Table 111. TIM2/TIM3 register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x24</td>
<td>TIMx_CNT</td>
<td>CNT[31:0]</td>
<td>CNT[30:16]</td>
<td>(TIM2 only, reserved on the other timers)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CNT[15:0]</td>
<td></td>
</tr>
<tr>
<td>0x28</td>
<td>TIMx_PSC</td>
<td></td>
<td>PSC[15:0]</td>
<td></td>
</tr>
<tr>
<td>0x2C</td>
<td>TIMx_ARR</td>
<td>ARR[31:16]</td>
<td>ARR[15:0]</td>
<td></td>
</tr>
<tr>
<td>0x30</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x34</td>
<td>TIMx_CCR1</td>
<td>CCR1[31:16]</td>
<td>CCR1[15:0]</td>
<td>(TIM2 only, reserved on the other timers)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x38</td>
<td>TIMx_CCR2</td>
<td>CCR2[31:16]</td>
<td>CCR2[15:0]</td>
<td>(TIM2 only, reserved on the other timers)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3C</td>
<td>TIMx_CCR3</td>
<td>CCR3[31:16]</td>
<td>CCR3[15:0]</td>
<td>(TIM2 only, reserved on the other timers)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x40</td>
<td>TIMx_CCR4</td>
<td>CCR4[31:16]</td>
<td>CCR4[15:0]</td>
<td>(TIM2 only, reserved on the other timers)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x48</td>
<td>TIMx_DCR</td>
<td></td>
<td>DBL[4:0]</td>
<td>DBA[4:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4C</td>
<td>TIMx_DMAR</td>
<td></td>
<td>DMAB[15:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x50</td>
<td>TIM2_OR1</td>
<td></td>
<td>OCREF_CLR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 111. TIM2/TIM3 register map and reset values  (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x50   | TIM3_OR1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x60   | TIM2_AF1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x60   | TIM3_AF1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x68   | TIM2_TISEL    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x68   | TIM3_TISEL    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.2 on page 54 for the register boundary addresses.
22 Basic timers (TIM6/TIM7)

22.1 TIM6/TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

The timers are completely independent, and do not share any resources.

22.2 TIM6/TIM7 main features

Basic timer (TIM6/TIM7) features include:
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 212. Basic timer block diagram
22.3 TIM6/TIM7 functional description

22.3.1 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 213* and *Figure 214* give some examples of the counter behavior when the prescaler ratio is changed on the fly.
Figure 213. Counter timing diagram with prescaler division change from 1 to 2

Figure 214. Counter timing diagram with prescaler division change from 1 to 4
22.3.2 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 215. Counter timing diagram, internal clock divided by 1
Figure 216. Counter timing diagram, internal clock divided by 2

- CK_PSC
- CNT_EN
- Timer clock = CK_CNT
- Counter register
- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)

Figure 217. Counter timing diagram, internal clock divided by 4

- CK_PSC
- CNT_EN
- Timer clock = CK_CNT
- Counter register
- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)
**Figure 218. Counter timing diagram, internal clock divided by N**

- **CK_PSC**
- Timer clock = CK_CNT
- Counter register: 1F 20 00
- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)

**Figure 219. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)**

- **CK_PSC**
- CEN
- Timer clock = CK_CNT
- Counter register: 31 32 33 34 35 36 00 01 02 03 04 05 06 07
- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)
- Auto-reload preload register: FF 36

Write a new value in TIMx_ARR
22.3.3 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register’s bit 31 (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

22.3.4 Clock source

The counter clock is provided by the Internal clock (CK_INT) source.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 221 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.
22.3.5 Debug mode

When the microcontroller enters the debug mode (Cortex®-M0+ core - halted), the TIMx counter either continues to work normally or stops, depending on the DBG_TIMx_STOP configuration bit in the DBG module. For more details, refer to Section 37.9.2: Debug support for timers, watchdog and I2C.

22.4 TIM6/TIM7 registers

Refer to Section 1.2 on page 50 for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

22.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

Address offset: 0x00
Reset value: 0x0000

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

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.
Bit 7 **ARPE**: Auto-reload preload enable

- 0: TIMx_ARR register is not buffered.
- 1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

- 0: Counter is not stopped at update event
- 1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

- 0: Any of the following events generates an update interrupt or DMA request if enabled.
  These events can be:
  - Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller
- 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

- 0: UEV enabled. The Update (UEV) event is generated by one of the following events:
  - Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

- 1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

- 0: Counter disabled
- 1: Counter enabled

**Note**: Gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.
### 22.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7)

Address offset: 0x04  
Reset value: 0x0000  

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

- **Bits 15:7** Reserved, must be kept at reset value.  
- **Bits 6:4** **MMS[2:0]: Master mode selection**  
  These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:  
  - **000: Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (TRGO). If reset is generated by the trigger input (slaves mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.  
  - **001: Enable** - the Counter enable signal, CNT_EN, is used as a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).  
  - **010: Update** - The update event is selected as a trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.  
  
  **Note:** The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.  

- **Bits 3:0** Reserved, must be kept at reset value.  

### 22.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)

Address offset: 0x0C  
Reset value: 0x0000  

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

- **Bits 15:9** Reserved, must be kept at reset value.  
  - **Bit 8** **UDE: Update DMA request enable**  
    0: Update DMA request disabled.  
    1: Update DMA request enabled.  

- **Bits 7:1** Reserved, must be kept at reset value.  
  - **Bit 0** **UIE: Update interrupt enable**  
    0: Update interrupt disabled.  
    1: Update interrupt enabled.
22.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)

Address offset: 0x10
Reset value: 0x0000

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

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow or underflow regarding the repetition counter value and if UDIS = 0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

22.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7)

Address offset: 0x14
Reset value: 0x0000

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

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

22.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)

Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CNT[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bit 31  **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16  Reserved, must be kept at reset value.

Bits 15:0  **CNT[15:0]**: Counter value

### 22.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7)
Address offset: 0x28
Reset value: 0x0000

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

Bits 15:0  **PSC[15:0]**: Prescaler value
The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

### 22.4.8 TIMx auto-reload register (TIMx.ARR)(x = 6 to 7)
Address offset: 0x2C
Reset value: 0xFFFF

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

Bits 15:0  **ARR[15:0]**: Prescaler value
ARR is the value to be loaded into the actual auto-reload register.
Refer to Section 22.3.1: Time-base unit on page 661 for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
### 22.4.9 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 112. TIMx register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | TIMx_CR1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | TIMx_CR2      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xC    | TIMx_DIER     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | TIMx_SR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x14   | TIMx_EGR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x18-  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x20   | TIMx_CNT      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | UECPT or Res. |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x28   | TIMx_PSC      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | PSC[15:0]     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x2C   | TIMx_ARR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | ARR[15:0]     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |

Refer to Section 2.2 on page 54 for the register boundary addresses.
23 General-purpose timers (TIM14)

23.1 TIM14 introduction

The TIM14 general-purpose timer consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM14 timer is completely independent, and does not share any resources.

23.2 TIM14 main features

23.2.1 TIM14 main features

The features of general-purpose timer TIM14 include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare
Figure 222. General-purpose timer block diagram (TIM14)

Notes:
- Preload registers transferred to active registers on U event according to control bit
- Event
- Interrupt & DMA output
23.3 **TIM14 functional description**

23.3.1 **Time-base unit**

The main block of the timer is a 16-bit up-counter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set.

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

**Prescaler description**

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 223* and *Figure 224* give some examples of the counter behavior when the prescaler ratio is changed on the fly.
Figure 223. Counter timing diagram with prescaler division change from 1 to 2

Figure 224. Counter timing diagram with prescaler division change from 1 to 4
23.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (by software) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 225. Counter timing diagram, internal clock divided by 1

![Counter timing diagram](image-url)
**Figure 226. Counter timing diagram, internal clock divided by 2**

- **CK_PSC**
- **CNT_EN**
- Timer clock = CK_CNT
- Counter register: 0034 0035 0036 0000 0001 0002 0003
- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)

**Figure 227. Counter timing diagram, internal clock divided by 4**

- **CK_PSC**
- **CNT_EN**
- Timer clock = CK_CNT
- Counter register: 0035 0036 0000 0001
- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)
Figure 228. Counter timing diagram, internal clock divided by N

Figure 229. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)
23.3.3 Clock selection

The counter clock can be provided by the following clock sources:
- Internal clock (CK_INT)

Internal clock source (CK_INT)

The internal clock source is the default clock source for TIM14.

*Figure 231* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.
### 23.3.4 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

*Figure 232 to Figure 234* give an overview of one capture/compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxF_Px) which can be used as the capture command. It is prescaled before the capture register (ICxPS).

*Figure 232. Capture/compare channel (example: channel 1 input stage)*

The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.
The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

23.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be
cleared by software by writing it to ‘0’ or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1[x] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to ‘01’ in the TIMx_CCMR1 register. As soon as CC1S becomes different from ‘00’, the channel is configured in input mode and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (by programming the ICxF bits in the TIMx_CCMRx register if the input is one of the TIx inputs). Let’s imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to ‘0011’ in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the TI1 channel by programming CC1P and CC1NP bits to ‘00’ in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_DIER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_EGR register.

When an input capture occurs:
- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

23.3.6 Forced output mode

In output mode (CCxS bits = ‘00’ in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, one just needs to write ‘0101’ in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=’0’ (OCx active high) => OCx is forced to high level.
The OCxREF signal can be forced low by writing the OCxM bits to ‘0100’ in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

23.3.7 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM='0000'), be set active (OCxM='0001'), be set inactive (OCxM='0010') or can toggle (OCxM='0011') on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = ‘0011’ to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = ‘0’ to disable preload register
   - Write CCxP = ‘0’ to select active high polarity
   - Write CCxE = ‘1’ to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 235.
23.3.8 PWM mode

Pulse Width Modulation mode allows to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

The OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. The OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CNT ≤ TIMx_CCRx.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. Figure 236 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.
23.3.9 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled using the CEN bit. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

\[ \text{CNT} < \text{CCR}_x = \text{ARR} \] (in particular, \( 0 < \text{CCR}_x \))

23.3.10 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMx_CNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.
23.3.11 Debug mode

When the microcontroller enters debug mode (Cortex®-M0+ core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module. For more details, refer to Section 37.9.2: Debug support for timers, watchdog and I²C.
23.4 TIM14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

23.4.1 TIM14 control register 1 (TIM14_CR1)

Address offset: 0x00

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-12</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>11</td>
<td>UIFREMAP</td>
<td>UIF status bit remapping</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>9-8</td>
<td>CKD[1:0]</td>
<td>Clock division</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00: tDTS = tCK_INT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01: tDTS = 2 × tCK_INT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10: tDTS = 4 × tCK_INT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11: Reserved</td>
</tr>
<tr>
<td>7</td>
<td>ARPE</td>
<td>Auto-reload preload enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: TIMx_ARR register is not buffered</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: TIMx_ARR register is buffered</td>
</tr>
<tr>
<td>6-4</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>3</td>
<td>OPM</td>
<td>One-pulse mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Counter is not stopped on the update event</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Counter stops counting on the next update event (clearing the CEN bit).</td>
</tr>
</tbody>
</table>

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 UIFREMAP: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 CKD[1:0]: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (T1x),
00: tDTS = tCK_INT
01: tDTS = 2 × tCK_INT
10: tDTS = 4 × tCK_INT
11: Reserved

Bit 7 ARPE: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 OPM: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).
23.4.2 TIM14 Interrupt enable register (TIM14_DIER)

Address offset: 0x0C

Reset value: 0x0000

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

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

23.4.3 TIM14 status register (TIM14_SR)

Address offset: 0x10

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
</tr>
</tbody>
</table>

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the update interrupt (UEV) sources.
0: Any of the following events generate an UEV if enabled:
   – Counter overflow
   – Setting the UG bit
1: Only counter overflow generates an UEV if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.
0: UEV enabled. An UEV is generated by one of the following events:
   – Counter overflow
   – Setting the UG bit.
Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled

*Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*
Bits 15:10  Reserved, must be kept at reset value.

Bit 9  **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:2  Reserved, must be kept at reset value.

Bit 1  **CC1IF**: Capture/compare 1 interrupt flag
If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value. It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.
If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred.
1: The counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).

Bit 0  **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
   – At overflow and if UDIS='0' in the TIMx_CR1 register.
   – When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS='0' and UDIS='0' in the TIMx_CR1 register.

### 23.4.4 TIM14 event generation register (TIM14_EGR)

Address offset: 0x14
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CC1G</td>
<td>UG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>
23.4.5 TIM14 capture/compare mode register 1 [alternate] (TIM14_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare
mode (next section). The direction of a channel is defined by configuring the corresponding
CCxS bits. All the other bits of this register have a different function in input and in output
mode.

**Input capture mode:**

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

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

Bits 7:4    IC1F[3:0]: Input capture 1 filter

This bit-field defines the frequency used to sample Ti1 input and the length of the digital filter
applied to Ti1. The digital filter is made of an event counter in which N consecutive events
are needed to validate a transition on the output:

0000: No filter, sampling is done at fDTS: 1000: fsampling = fDTS/8, N=6
0001: fsampling = fCK_INT, N=21001: fSampling = fDTS/8, N=8
0010: fsampling = fCK_INT, N=41010: fSampling = fDTS/16, N=5
0011: fsampling = fCK_INT, N=81011: fSampling = fDTS/16, N=6
0100: fsampling = fDTS/2, N=61000: fSampling = fDTS/16, N=8
0101: fsampling = fDTS/2, N=61011: fSampling = fDTS/32, N=5
0110: fsampling = fDTS/4, N=61100: fSampling = fDTS/32, N=6
0111: fsampling = fDTS/4, N=81111: fSampling = fDTS/32, N=8
Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: Reserved
11: Reserved

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).*

---

**23.4.6 TIM14 capture/compare mode register 1 [alternate] (TIM14_CCMR1)**

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Output compare mode:**

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

**OC1M[2:0]**: OC1PE OC1FE CC1S[1:0]
Bits 31:17  Reserved, must be kept at reset value.

Bits 15:7  Reserved, must be kept at reset value.

Bits 16, 6:4  OC1M[3:0]: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas OC1 active level depends on CC1P bit.

0000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active

Others: Reserved

Note: In PWM mode 1 or 2, the OCREF level changes when the result of the comparison changes or when the output compare mode switches from frozen to PWM mode.

Bit 3  OC1PE: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2  OC1FE: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. OC is then set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0  CC1S[1:0]: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

Others: Reserved.

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).
23.4.7  TIM14 capture/compare enable register (TIM14_CCER)

Address offset: 0x20
Reset value: 0x0000

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

Bits 15:4  Reserved, must be kept at reset value.

Bit 3  CC1NP: Capture/Compare 1 complementary output Polarity.
CC1 channel configured as output: CC1NP must be kept cleared.
CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define TI1FP1 polarity (refer to CC1P description).

Bit 2  Reserved, must be kept at reset value.

Bit 1  CC1P: Capture/Compare 1 output Polarity.
**CC1 channel configured as output:**
0: OC1 active high
1: OC1 active low

**CC1 channel configured as input:**
The CC1P and CC1NP bits select TI1FP1 polarity for capture operations.
00: noninverted/rising edge
Circuit is sensitive to TI1FP1 rising edge (capture mode), TI1FP1 is not inverted.
01: inverted/falling edge
Circuit is sensitive to TI1FP1 falling edge (capture mode), TI1FP1 is inverted.
10: reserved, do not use this configuration.
11: noninverted/both edges
Circuit is sensitive to both TI1FP1 rising and falling edges (capture mode), TI1FP1 is not inverted.

Bit 0  CC1E: Capture/Compare 1 output enable.
**CC1 channel configured as output:**
0: Off - OC1 is not active
1: On - OC1 signal is output on the corresponding output pin

**CC1 channel configured as input:**
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled
1: Capture enabled

**Table 113. Output control bit for standard OCx channels**

<table>
<thead>
<tr>
<th>CCxE bit</th>
<th>OCx output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output Disabled (OCx='0', OCx_EN='0')</td>
</tr>
<tr>
<td>1</td>
<td>OCx=OCxREF + Polarity, OCx_EN='1'</td>
</tr>
</tbody>
</table>

**Note:**  The state of the external I/O pins connected to the standard OCx channels depends on the OCx channel state and the GPIO registers.
23.4.8 TIM14 counter (TIM14_CNT)
Address offset: 0x24
Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

23.4.9 TIM14 prescaler (TIM14_PSC)
Address offset: 0x28
Reset value: 0x0000

<table>
<thead>
<tr>
<th>PSC[15:0]</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
</tr>
</tbody>
</table>

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency $f_{\text{CK\_CNT}}$ is equal to $f_{\text{CK\_PSC}} / (\text{PSC}[15:0] + 1)$.
PSC contains the value to be loaded in the active prescaler register at each update event.
(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

23.4.10 TIM14 auto-reload register (TIM14_ARR)
Address offset: 0x2C
Reset value: 0xFFFF

<table>
<thead>
<tr>
<th>ARR[15:0]</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
</tr>
</tbody>
</table>

Bits 15:0 **ARR[15:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to Section 23.3.1: Time-base unit on page 675 for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
23.4.11 TIM14 capture/compare register 1 (TIM14_CCR1)

Address offset: 0x34
Reset value: 0x0000

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

Bits 15:0 CCR1[15:0]: Capture/Compare 1 value

- **If channel CC1 is configured as output:**
  - CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
  - It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
  - The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

- **If channel CC1 is configured as input:**
  - CCR1 is the counter value transferred by the last input capture 1 event (IC1).

23.4.12 TIM14 timer input selection register (TIM14_TISEL)

Address offset: 0x68
Reset value: 0x0000

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

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 T1SEL[3:0]: selects T1[0] to T1[15] input

- 0000: TIM14_CH1 input
- 0001: RTC CLK
- 0010: HSE/32
- 0011: MCO
- Others: Reserved
### 23.4.13 TIM14 register map

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

| Offset   | Register name   | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00     | TIMx_CR1        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04     | Reserved        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C     | TIMx_DIER       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10     | TIMx_SR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x14     | TIMx_EGR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x18     | TIMx_CCMR1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Output compare  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | mode            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x1C     | Reserved        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x20     | TIMx_CCER       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x24     | TIMx_CNT        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | UFC/PY         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x28     | TIMx_PSC        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | PSC[15:0]       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2C     | TIMx_ARR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | ARR[15:0]       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x30     | Reserved        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Table 114. TIM14 register map and reset values
Table 114. TIM14 register map and reset values (continued)

| Offset   | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x34     | TIMx_CCR1     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x38 to 0x64 | Reserved    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x68     | TIM14_TISEL   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to *Section 2.2 on page 54* for the register boundary addresses.
24 General-purpose timers (TIM15/TIM16/TIM17)

24.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in Section 24.4.22: Timer synchronization (TIM15).

24.2 TIM15 main features

TIM15 includes the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also "on the fly") the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)
24.3 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input
Figure 237. TIM15 block diagram

1. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to Section 5.2.8: Clock security system (CSS)
   - A PVD output
   - SRAM parity error signal
   - Cortex®-M0+ LOCKUP (Hardfault) output
   - COMP output

Notes:
- Preload registers transferred to active registers on U event according to control bit
- Event
- Interrupt & DMA output
1. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to Section 5.2.8: Clock security system (CSS)
   - A PVD output
   - SRAM parity error signal
   - Cortex®-M0+ LOCKUP (Hardfault) output
   - COMP output
24.4 TIM15/TIM16/TIM17 functional description

24.4.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 239* and *Figure 240* give some examples of the counter behavior when the prescaler ratio is changed on the fly:
Figure 239. Counter timing diagram with prescaler division change from 1 to 2

Figure 240. Counter timing diagram with prescaler division change from 1 to 4
24.4.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.
Figure 241. Counter timing diagram, internal clock divided by 1

Figure 242. Counter timing diagram, internal clock divided by 2
Figure 243. Counter timing diagram, internal clock divided by 4

Figure 244. Counter timing diagram, internal clock divided by N
Figure 245. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

Figure 246. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)
24.4.3 Repetition counter

Section 24.4.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to Figure 247). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.
24.4.4 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin
- Internal trigger inputs (ITRx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as the prescaler for TIM15. Refer to Using one timer as prescaler for another timer on page 625 for more details.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed...
only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

*Figure 248* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

**Figure 248. Control circuit in normal mode, internal clock divided by 1**

![Control circuit in normal mode, internal clock divided by 1](image)

**External clock source mode 1**

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

*Figure 249. TI2 external clock connection example*

![TI2 external clock connection example](image)

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:
1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so it does not need to be configured.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

**Figure 250. Control circuit in external clock mode 1**

24.4.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

*Figure 251 to Figure 254* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).
Figure 251. Capture/compare channel (example: channel 1 input stage)

The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 252. Capture/compare channel 1 main circuit
The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 24.4.6 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was
already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:
- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

*Note:* IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

### 24.4.7 PWM input mode (only for TIM15)

This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.
For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper TI1[x] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to ‘1’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

**Figure 255. PWM input mode timing**

1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

### 24.4.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.
Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

### 24.4.9 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

### Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 256.
24.4.10 PWM mode

Pulse Width Modulation mode allows to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘110’ (PWM mode 1) or ‘111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to Upcounting mode on page 705.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at
If the compare value is 0 then OCxRef is held at '0'. Figure 257 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

**Figure 257. Edge-aligned PWM waveforms (ARR=8)**

<table>
<thead>
<tr>
<th>Counter register</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCRx=4 OCXREF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx=4 CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx=8 OCXREF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx=8 CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx&gt;8 OCXREF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'1'</td>
<td></td>
</tr>
<tr>
<td>CCRx&gt;8 CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx=0 OCXREF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'0'</td>
</tr>
<tr>
<td>CCRx=0 CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 258** represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
24.4.12 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to Table 119: Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) on page 774 for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a
reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

**Figure 259. Complementary output with dead-time insertion.**

![Diagram](MS31095V1)

**Figure 260. Dead-time waveforms with delay greater than the negative pulse.**

![Diagram](MS31096V1)
Figure 261. Dead-time waveforms with delay greater than the positive pulse.

The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to Section 24.6.14: TIMx break and dead-time register (TIMx_BDTR) (x = 16 to 17) on page 777 for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

24.4.13 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM15/TIM16/TIM17 timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.
The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows to enable/disable the outputs by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shutdown level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to Table 117: Output control bits for complementary OCx and OCxN channels with break feature (TIM15) on page 753 for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

A programmable filter (BKF[3:0] bits in the TIMx_BDTR register allows to filter out spurious events.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 register.

The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source:
  - the output from a comparator, with polarity selection and optional digital filtering
  - A system break:
    - the Cortex®-M0+ LOCKUP output
    - the PVD output
    - the SRAM parity error signal
    - a Flash ECC error
    - a clock failure event generated by the CSS detector
Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the AFIO controller (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the AFIO controller) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their
active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).

- If OSSI=0 then the timer releases the enable outputs (taken over by the AFIO controller which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set.

- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows to freeze the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to Section 24.6.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 777. The LOCK bits can be written only once after an MCU reset.

The Figure 263 shows an example of behavior of the outputs in response to a break.
Figure 263. Output behavior in response to a break

- OCxREF
- OCx (OCxN not implemented, CCxP=0, OISx=1)
- OCx (OCxN not implemented, CCxP=0, OISx=0)
- OCx (OCxN not implemented, CCxP=1, OISx=1)
- OCx (OCxN not implemented, CCxP=1, OISx=0)
- OCx
- OCxN (CCxE=1, CCxP=0, OISx=0, CCxNE=1, CCxNP=0, OISxN=1)
- OCxN (CCxE=1, CCxP=0, OISx=1, CCxNE=1, CCxNP=0, OISxN=1)
- OCxN (CCxE=1, CCxP=0, OISx=0, CCxNE=0, CCxNP=0, OISxN=1)
- OCxN (CCxE=1, CCxP=0, OISx=1, CCxNE=0, CCxNP=0, OISxN=0)
- OCxN (CCxE=1, CCxP=0, OISx=0, CCxNE=0, CCxNP=0, OISxN=0 or OISx=OISxN=1)
24.4.14 Bidirectional break inputs

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on Figure 264.

They allow to have:
- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (BG) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:
- The break input path is always active: a break event will be active even if the BKDSRM bit is set and the open drain control is released. This will prevent the PWM output to be re-started as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see Table 115)

Table 115. Break protection disarming conditions

<table>
<thead>
<tr>
<th>MOE</th>
<th>BKDIR</th>
<th>BKDSRM</th>
<th>Break protection state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Disarmed</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Armed</td>
</tr>
</tbody>
</table>

Arming and re-arming break circuitry

The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).
The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

**Figure 264. Output redirection**
24.4.15 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- $CNT < CCRx \leq ARR$ (in particular, $0 < CCRx$)
Figure 265. Example of one pulse mode.

For example one may want to generate a positive pulse on OC1 with a length of $t_{\text{PULSE}}$ and after a delay of $t_{\text{DELAY}}$ as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:
1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC 2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='00110' in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to ‘110’ in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The $t_{\text{DELAY}}$ is defined by the value written in the TIMx_CCR1 register.
- The $t_{\text{PULSE}}$ is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).

Let's say one want to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=’1’ in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0’ in this example.

Since only 1 pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).
Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay $t_{\text{DELAY min}}$ we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

24.4.16 Retriggerable one pulse mode (OPM) (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in Section 24.4.15:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

24.4.17 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMx_CNT[31]). This allows to
atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.
24.4.18 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins TIMx_CH1 and TIMx_CH2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in Figure 267.

**Figure 267. Measuring time interval between edges on 2 signals**
24.4.19 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P='0' and CC1NP='0' in the TIMx_CCER register to validate the polarity (and detect rising edges only).

2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

![Figure 268. Control circuit in reset mode](image-url)
Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP = ‘0’ in the TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 269. Control circuit in gated mode
Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P='1' and CC2NP='0' in the TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS=110 in the TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in the TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

24.4.20 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

24.4.21 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to re-program several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.
The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address, i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,

For example, the timer DMA burst feature could be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:
1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   – Number of data to transfer = 3 (See note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.
24.4.22 Timer synchronization (TIM15)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to Section 21.3.19: Timer synchronization for details.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

24.4.23 Debug mode

When the microcontroller enters debug mode (Cortex®-M0+ core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module. For more details, refer to Section 37.9.2: Debug support for timers, watchdog and I^2C.

For safety purposes, when the counter is stopped (DBG_TIMx_STOP = 1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.
24.5 **TIM15 registers**

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

24.5.1 **TIM15 control register 1 (TIM15_CR1)**

Address offset: 0x00

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>UIFREMAP</td>
<td>CKD[1:0]</td>
<td>ARPE</td>
<td>OPM</td>
<td>URS</td>
<td>UDIS</td>
<td>CEN</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (tDTS) used by the dead-time generators and the digital filters (TIx)

00: tDTS = tCK_INT
01: tDTS = 2*tCK_INT
10: tDTS = 4*tCK_INT
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)
Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
*Note*: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

### 24.5.2 TIM15 control register 2 (TIM15_CR2)

Address offset: 0x04
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td>OIS2: Output idle state 2 (OC2 output)</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>OC2=0 when MOE=0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>OC2=1 when MOE=0</td>
<td>1</td>
</tr>
</tbody>
</table>

*Note*: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIMx_BKR register).

Bit 9 **OIS1**: Output Idle state 1 (OC1N output)
0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0

*Note*: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)
0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

*Note*: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).
Bit 7 **TI1S**: TI1 selection
- 0: The TIMx_CH1 pin is connected to TI1 input
- 1: The TIMx_CH1, CH2 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
- 000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
- 001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
- 010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
- 011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (TRGO).
- 100: **Compare** - OC1REF signal is used as trigger output (TRGO).
- 101: **Compare** - OC2REF signal is used as trigger output (TRGO).

Bit 3 **CCDS**: Capture/compare DMA selection
- 0: CCx DMA request sent when CCx event occurs
- 1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection
- 0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.
- 1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

*Note: This bit acts only on channels that have a complementary output.*

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
- 0: CCxE, CCxNE and OCxM bits are not preloaded
- 1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).

*Note: This bit acts only on channels that have a complementary output.*
### 24.5.3 TIM15 slave mode control register (TIM15_SMCR)

Address offset: 0x08  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15</td>
<td>Bit 7 MSM: Master/slave mode</td>
</tr>
<tr>
<td>14</td>
<td>0: No action</td>
</tr>
<tr>
<td>13</td>
<td>1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.</td>
</tr>
<tr>
<td>12</td>
<td>Bits 21, 20, 6, 5, 4 TS[4:0]: Trigger selection</td>
</tr>
<tr>
<td>11</td>
<td>This bit field selects the trigger input to be used to synchronize the counter.</td>
</tr>
<tr>
<td>10</td>
<td>00000: Internal Trigger 0 (ITR0)</td>
</tr>
<tr>
<td>9</td>
<td>00001: Internal Trigger 1 (ITR1)</td>
</tr>
<tr>
<td>8</td>
<td>00010: Internal Trigger 2 (ITR2)</td>
</tr>
<tr>
<td>7</td>
<td>00011: Internal Trigger 3 (ITR3)</td>
</tr>
<tr>
<td>6</td>
<td>00100: TI1 Edge Detector (TI1F_ED)</td>
</tr>
<tr>
<td>5</td>
<td>00101: Filtered Timer Input 1 (TI1FP1)</td>
</tr>
<tr>
<td>4</td>
<td>00110: Filtered Timer Input 2 (TI2FP2)</td>
</tr>
<tr>
<td>3</td>
<td>Other: Reserved</td>
</tr>
<tr>
<td>2</td>
<td>See Table 116: TIMx Internal trigger connection on page 743 for more details on ITRx meaning for each Timer.</td>
</tr>
<tr>
<td>1</td>
<td>Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.</td>
</tr>
<tr>
<td>0</td>
<td>Bit 3 Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
24.5.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Slave TIM</th>
<th>ITR0 (TS = 00000)</th>
<th>ITR1 (TS = 00001)</th>
<th>ITR2 (TS = 00010)</th>
<th>ITR3 (TS = 00011)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM15</td>
<td>TIM2</td>
<td>TIM3</td>
<td>TIM16_OC1</td>
<td>TIM17_OC1</td>
</tr>
</tbody>
</table>

- **TDE**: Trigger DMA request enable
  - 0: Trigger DMA request disabled
  - 1: Trigger DMA request enabled

- **COMDE**: COM DMA request enable
  - 0: COM DMA request disabled
  - 1: COM DMA request enabled
24.5.5 TIM15 status register (TIM15_SR)

Address offset: 0x10
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15 (CC1OF)</th>
<th>Bit 14 (CC0OF)</th>
<th>Bit 13 (BIIF)</th>
<th>Bit 12 (TIF)</th>
<th>Bit 11 (COMIF)</th>
<th>Bit 10 (CC2IF)</th>
<th>Bit 9 (CC1IF)</th>
<th>Bit 8 (UIF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
   Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
   This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
   0: No overcapture has been detected
   1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag
   This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
   0: No break event occurred
   1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag
   This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
   0: No trigger event occurred
   1: Trigger interrupt pending

Bit 5 **COMIF**: COM interrupt flag
   This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
   0: No COM event occurred
   1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.
Bit 2 CC2IF: Capture/Compare 2 interrupt flag
   refer to CC1IF description

Bit 1 CC1IF: Capture/Compare 1 interrupt flag
   If channel CC1 is configured as output: This flag is set by hardware when the counter
   matches the compare value. It is cleared by software.
   0: No match.
   1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register.
      When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF
      bit goes high on the counter overflow.
   If channel CC1 is configured as input: This bit is set by hardware on a capture. It is
   cleared by software or by reading the TIMx_CCR1 register.
   0: No input capture occurred
   1: The counter value has been captured in TIMx_CCR1 register (An edge has been
      detected on IC1 which matches the selected polarity)

Bit 0 UIF: Update interrupt flag
   This bit is set by hardware on an update event. It is cleared by software.
   0: No update occurred.
   1: Update interrupt pending. This bit is set by hardware when the registers are updated:
      – At overflow regarding the repetition counter value (update if repetition counter = 0) and if
      the UDIS=0 in the TIMx_CR1 register.
      – When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0
      and UDIS=0 in the TIMx_CR1 register.
      – When CNT is reinitialized by a trigger event (refer to Section 24.5.3: TIM15 slave mode
      control register (TIM15_SMCR)), if URS=0 and UDIS=0 in the TIMx_CR1 register.

24.5.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14
Reset value: 0x0000

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

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 BG: Break generation
   This bit is set by software in order to generate an event, it is automatically cleared by
   hardware.
   0: No action
   1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or
      DMA transfer can occur if enabled.

Bit 6 TG: Trigger generation
   This bit is set by software in order to generate an event, it is automatically cleared by
   hardware.
   0: No action
   1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if
      enabled
Bit 5  **COMG**: Capture/Compare control update generation
   This bit can be set by software, it is automatically cleared by hardware.
   0: No action
   1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits
   
   Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2  **CC2G**: Capture/Compare 2 generation
   Refer to CC1G description

Bit 1  **CC1G**: Capture/Compare 1 generation
   This bit is set by software in order to generate an event, it is automatically cleared by hardware.
   0: No action
   1: A capture/compare event is generated on channel 1:
   
   **If channel CC1 is configured as output:**
   CC1IF flag is set, corresponding interrupt or DMA request is sent if enabled.
   
   **If channel CC1 is configured as input:**
   The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0  **UG**: Update generation
   This bit can be set by software, it is automatically cleared by hardware.
   0: No action.
   1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

24.5.7  **TIM15 capture/compare mode register 1 [alternate]**
   (TIM15_CCMR1)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Input capture mode:**

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

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

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:12  **IC2F[3:0]**: Input capture 2 filter

Bits 11:10  **IC2PSC[1:0]**: Input capture 2 prescaler
Bits 9:8 CC2S[1:0]: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 IC1F[3:0]: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:
0000: No filter, sampling is done at fDTS
0001: fSAMPLING= fCK_INT, N=2
0010: fSAMPLING= fCK_INT, N=4
0011: fSAMPLING= fCK_INT, N=8
0100: fSAMPLING= fDTS/2, N=6
0101: fSAMPLING= fDTS/2, N=8
0110: fSAMPLING= fDTS/4, N=6
0111: fSAMPLING= fDTS/4, N=8
1000: fSAMPLING= fDTS/8, N=6
1001: fSAMPLING= fDTS/8, N=8
1010: fSAMPLING= fDTS/16, N=5
1011: fSAMPLING= fDTS/16, N=6
1100: fSAMPLING= fDTS/32, N=5
1101: fSAMPLING= fDTS/32, N=6
1110: fSAMPLING= fDTS/32, N=8
1111: fSAMPLING= fDTS/32, N=8

Bits 3:2 IC1PSC[1:0]: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 CC1S[1:0]: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

24.5.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the
corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Output compare mode:**

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

Bits 31:25  Reserved, must be kept at reset value.
Bits 23:17  Reserved, must be kept at reset value.
Bit 15    Reserved, must be kept at reset value.

**OC2M[3:0]:** Output Compare 2 mode
- Bit 11 **OC2PE**: Output Compare 2 preload enable
- Bit 10 **OC2FE**: Output Compare 2 fast enable

**CC2S[1:0]:** Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
- 00: CC2 channel is configured as output.
- 01: CC2 channel is configured as input, IC2 is mapped on TI2.
- 10: CC2 channel is configured as input, IC2 is mapped on TI1.
- 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

**Note:** CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7    Reserved, must be kept at reset value.
Bits 16, 6:4 **OC1M[3:0]: Output Compare 1 mode**

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Reserved

1111: Reserved

**Note:**

1. These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

2. In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

3. On channels that have a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.
Bit 3 **OC1PE**: Output Compare 1 preload enable
0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

**Note:** 1: These bits cannot be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).
2: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output Compare 1 fast enable
This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, IC1 is mapped on TI1.
10: CC1 channel is configured as input, IC1 is mapped on TI2.
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

**Note:** CC1S bits are writable only when the channel is OFF (CC1E = ‘0’ in TIMx_CCER).

### 24.5.9 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

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

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description
Bit 3  **CC1NP**: Capture/Compare 1 complementary output polarity
- CC1 channel configured as output:
  - 0: OC1N active high
  - 1: OC1N active low
- CC1 channel configured as input:
  - This bit is used in conjunction with CC1P to define the polarity of Ti1FP1 and Ti2FP1. Refer to CC1P description.

  **Note**: 1. *This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).*
  2. *On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 2  **CC1NE**: Capture/Compare 1 complementary output enable
- 0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
- 1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1  **CC1P**: Capture/Compare 1 output polarity
- CC1 channel configured as output:
  - 0: OC1 active high
  - 1: OC1 active low
- **CC1 channel configured as input**: The CC1NP/CC1P bits select the polarity of Ti1FP1 and Ti2FP1 for trigger or capture operations.
  - 00: non-inverted/rising edge. The circuit is sensitive to Ti1FP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), Ti1FP1 is not inverted (trigger operation in gated mode).
  - 01: inverted/falling edge. The circuit is sensitive to Ti1FP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), Ti1FP1 is inverted (trigger operation in gated mode).
  - 10: reserved, do not use this configuration.
  - 11: non-inverted/both edges. The circuit is sensitive to both Ti1FP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), Ti1FP1 is not inverted (trigger operation in gated mode).

  **Note**: 1. *This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*
  2. *On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 0  **CC1E**: Capture/Compare 1 output enable
- **CC1 channel configured as output**: 
  - 0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
  - 1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
- **CC1 channel configured as input**: This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
  - 0: Capture disabled
  - 1: Capture enabled
### Table 117. Output control bits for complementary OCx and OCxN channels with break feature (TIM15)

<table>
<thead>
<tr>
<th>MOE bit</th>
<th>OSSI bit</th>
<th>OSSR bit</th>
<th>CCxE bit</th>
<th>CCxNE bit</th>
<th>OCx output state</th>
<th>OCxN output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Output Disabled (not driven by the timer: Hi-Z) OCx=0 OCxN=0, OCxN_EN=0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Output Disabled (not driven by the timer: Hi-Z) OCx=0</td>
<td>OCxREF + Polarity OCxN=OCxREF XOR CCxNP</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>OCxREF + Polarity OCx=OCxREF XOR CCxP OCxN=0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>OCREF + Polarity + dead-time Complementary to OCREF (not OCREF) + Polarity + dead-time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Off-State (output enabled with inactive state) OCx=CCxP</td>
<td>OCxREF + Polarity OCxN=OCxREF XOR CCxNP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>OCxREF + Polarity OCx=OCxREF XOR CCxP OCxN=CCxNP OCxN_EN=1</td>
<td>Off-State (output enabled with inactive state)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Output disabled (not driven by the timer anymore). The output state is defined by the GPIO controller and can be High, Low or Hi-Z.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP OCxN=CCxNP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Off-State (output enabled with inactive state) Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCx and OCxN both in active state</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

**Note:** The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and AFIO registers.
24.5.10 TIM15 counter (TIM15_CNT)
Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UIFCPY: UIF Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This bit is a read-only copy of the UIF bit in the TIMx_ISR register.</td>
</tr>
</tbody>
</table>

| Bit 30:16 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>CNT[15:0]: Counter value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>Counter value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UIF</th>
<th>Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

24.5.11 TIM15 prescaler (TIM15_PSC)
Address offset: 0x28
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>PSC[15:0]: Prescaler value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>Prescaler value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>PSC[15:0]: Prescaler value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>Prescaler value</td>
</tr>
</tbody>
</table>

24.5.12 TIM15 auto-reload register (TIM15_ARR)
Address offset: 0x2C
Reset value: 0xFFFF

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>ARR[15:0]: Auto-reload value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>Auto-reload value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>ARR[15:0]: Auto-reload value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>Auto-reload value</td>
</tr>
</tbody>
</table>
24.5.13  TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits 15:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td>REP[7:0]: Repetition counter value</td>
</tr>
</tbody>
</table>

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

24.5.14  TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>CCR1[15:0]: Capture/Compare 1 value</th>
</tr>
</thead>
</table>

**If channel CC1 is configured as output:**
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

**If channel CC1 is configured as input:**
CCR1 is the counter value transferred by the last input capture 1 event (IC1).
### 24.5.15 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>CCR2[15:0] Capture/Compare 2 value</td>
</tr>
<tr>
<td></td>
<td><strong>If channel CC2 is configured as output:</strong></td>
</tr>
<tr>
<td></td>
<td>- CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).</td>
</tr>
<tr>
<td></td>
<td>- It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.</td>
</tr>
<tr>
<td></td>
<td>- The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.</td>
</tr>
<tr>
<td></td>
<td><strong>If channel CC2 is configured as input:</strong></td>
</tr>
<tr>
<td></td>
<td>- CCR2 is the counter value transferred by the last input capture 2 event (IC2).</td>
</tr>
</tbody>
</table>

### 24.5.16 TIM15 break and dead-time register (TIM15_BDTR)

Address offset: 0x44  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td><strong>BKBID</strong>: Break Bidirectional</td>
</tr>
<tr>
<td></td>
<td>- 0: Break input BRK in input mode</td>
</tr>
<tr>
<td></td>
<td>- 1: Break input BRK in bidirectional mode</td>
</tr>
<tr>
<td></td>
<td>In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.</td>
</tr>
</tbody>
</table>
Bit 27  Reserved, must be kept at reset value.

Bit 26  **BKDSRM**: Break Disarm
- 0: Break input BRK is armed
- 1: Break input BRK is disarmed
  
  This bit is cleared by hardware when no break source is active.

  The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

  **Note**: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20  Reserved, must be kept at reset value.

Bits 19:16  **BKF[3:0]**: Break filter
This bit-field defines the frequency used to sample the BRK input signal and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

- 0000: No filter, BRK acts asynchronously
- 0001: $f_{SAMPLING} = f_{CK\ INT}$, N=2
- 0010: $f_{SAMPLING} = f_{CK\ INT}$, N=4
- 0011: $f_{SAMPLING} = f_{CK\ INT}$, N=8
- 0100: $f_{SAMPLING} = f_{DTS}/2$, N=6
- 0101: $f_{SAMPLING} = f_{DTS}/2$, N=8
- 0110: $f_{SAMPLING} = f_{DTS}/4$, N=6
- 0111: $f_{SAMPLING} = f_{DTS}/4$, N=8
- 1000: $f_{SAMPLING} = f_{DTS}/8$, N=6
- 1001: $f_{SAMPLING} = f_{DTS}/8$, N=8
- 1010: $f_{SAMPLING} = f_{DTS}/16$, N=5
- 1011: $f_{SAMPLING} = f_{DTS}/16$, N=6
- 1100: $f_{SAMPLING} = f_{DTS}/16$, N=8
- 1101: $f_{SAMPLING} = f_{DTS}/32$, N=5
- 1110: $f_{SAMPLING} = f_{DTS}/32$, N=6
- 1111: $f_{SAMPLING} = f_{DTS}/32$, N=8

  **Note**: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15  **MOE**: Main output enable
This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

- 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
- 1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)

  See OC/OCN enable description for more details (Section 24.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 751).

Bit 14  **AOE**: Automatic output enable
- 0: MOE can be set only by software
- 1: MOE can be set by software or automatically at the next update event (if the break input is not active)

  **Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 13 **BKP**: Break polarity
   - 0: Break input BRK is active low
   - 1: Break input BRK is active high

   **Note:**
   1: *This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*
   2: *Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 12 **BKE**: Break enable
   - 0: Break inputs (BRK and CCS clock failure event) disabled
   - 1: Break inputs (BRK and CCS clock failure event) enabled

   This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

   **Note:** *Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 11 **OSSR**: Off-state selection for Run mode
   This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

   See OC/OCN enable description for more details (Section 24.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 751).
   - 0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)
   - 1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

   **Note:** *This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 10 **OSSI**: Off-state selection for Idle mode
   This bit is used when MOE=0 on channels configured as outputs.

   See OC/OCN enable description for more details (Section 24.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 751).
   - 0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)
   - 1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1 (OC/OCN enable output signal=1)

   **Note:** *This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).*

Bits 9:8 **LOCK[1:0]**: Lock configuration
   These bits offer a write protection against software errors.
   - 00: LOCK OFF - No bit is write protected
   - 01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written
   - 10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
   - 11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

   **Note:** *The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.*
Bits 7:0 **DTG[7:0]: Dead-time generator setup**

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

- $DTG[7:5]=0xx \Rightarrow DT=DTG[7:0]\times t_{dtg}$ with $t_{dtg}=t_{DTS}$
- $DTG[7:5]=10x \Rightarrow DT=(64+DTG[5:0])\times t_{dtg}$ with $t_{dtg}=2t_{DTS}$
- $DTG[7:5]=110 \Rightarrow DT=(32+DTG[4:0])\times t_{dtg}$ with $T_{dtg}=8t_{DTS}$
- $DTG[7:5]=111 \Rightarrow DT=(32+DTG[4:0])\times t_{dtg}$ with $T_{dtg}=16t_{DTS}$

Example if $T_{DTS}=125\text{ns}$ ($8\text{MHz}$), dead-time possible values are:

- 0 to 15875 ns by 125 ns steps,
- 16 $\mu$s to 31750 ns by 250 ns steps,
- 32 $\mu$s to 63 $\mu$s by 1 $\mu$s steps,
- 64 $\mu$s to 126 $\mu$s by 2 $\mu$s steps

*Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

### 24.5.17 TIM15 DMA control register (TIM15_DCR)

**Address offset**: 0x48

**Reset value**: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Read/Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:13</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:8</td>
<td><strong>DBL[4:0]: DMA burst length</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00000: 1 transfer,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00001: 2 transfers,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00010: 3 transfers,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:5</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4:0</td>
<td><strong>DBA[4:0]: DMA base address</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00000: TIMx_CR1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00001: TIMx_CR2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00010: TIMx_SMCR,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 24.5.18 TIM15 DMA address for full transfer (TIM15_DMAR)

**Address offset**: 0x4C

**Reset value**: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Read/Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td><strong>DMAB[15:0]:</strong> DMA address</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(when read/write access are done through the TIMx_DMAR address).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00000: TIMx_CR1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00001: TIMx_CR2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00010: TIMx_SMCR,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
24.5.19 TIM15 alternate register 1 (TIM15_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 BKCM2P: BRK COMP2 input polarity
This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP2 input is active low
1: COMP2 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 BKCM1P: BRK COMP1 input polarity
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP1 input is active low
1: COMP1 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 BKINP: BRK BKIN input polarity
This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: BKIN input is active low
1: BKIN input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 BKCM2E: BRK COMP2 enable
This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.
0: COMP2 input disabled
1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 1 **BKCMP1E**: BRK COMP1 enable
This bit enables the COMP1 for the timer’s BRK input. COMP1 output is ‘ORed’ with the other BRK sources.
0: COMP1 input disabled
1: COMP1 input enabled

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable
This bit enables the BKIN alternate function input for the timer’s BRK input. BKIN input is ‘ORed’ with the other BRK sources.
0: BKIN input disabled
1: BKIN input enabled

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

### 24.5.20 TIM15 input selection register (TIM15_TISEL)

Address offset: 0x68
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects Ti2[0] to Ti2[15] input
- 0000: TIM15_CH2 input
- 0001: TIM2_IC2
- 0010: TIM3_IC2
- Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects Ti1[0] to Ti1[15] input
- 0000: TIM15_CH1 input
- 0001: TIM2_IC1
- 0010: TIM3_IC1
- Others: Reserved
### 24.5.21 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

#### Table 118. TIM15 register map and reset values

| Offset | Register name | Offset name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | TIM15_CR1     |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        | Reset value   |             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x04   | TIM15_CR2     |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x08   | TIM15_SMCR    |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 1 |
|        | Reset value   |             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x0C   | TIM15_DIER    |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        | Reset value   |             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x10   | TIM15_SR      |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        | Reset value   |             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x14   | TIM15_EGR     |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        | Reset value   |             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x18   | TIM15_CCMR1   |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
| Output Compare mode | OC2M[3]   |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        | Reset value   |             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x1C   | TIM15_CCMR1   |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
| Input Capture mode | IC2F[3:0] |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        | Reset value   |             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x20   | TIM15_CCER    |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0 |
|        | Reset value   |             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
### Table 118. TIM15 register map and reset values (continued)

| Offset | Register name  | Length | Reset value | Bit 31 | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------|----------------|--------|-------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0x24   | TIM15_CNT      | 16     | 0           | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |        |
| 0x28   | TIM15_PSC      | 16     | 0           | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |        |
| 0x2C   | TIM15_ARR      | 16     | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 |        |
| 0x30   | TIM15_RCR      | 16     | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |        |
| 0x34   | TIM15_CCR1     | 16     | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |        |
| 0x38   | TIM15_CCR2     | 16     | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |        |
| 0x44   | TIM15_BDTR     | 16     | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |        |
| 0x48   | TIM15_DCR      | 16     | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |        |
| 0x4C   | TIM15_DMAR     | 16     | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |        |
| 0x60   | TIM15_AF1      | 16     | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |        |
| 0x68   | TIM15_TISEL    | 16     | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |        |

Refer to Section 2.2 on page 54 for the register boundary addresses.
24.6 TIM16/TIM17 registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

24.6.1 TIMx control register 1 (TIMx_CR1) (x = 16 to 17)

Address offset: 0x00
Reset value: 0x0000

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

Bits 15:12 Reserved, must be kept at reset value.

- **Bit 11** UIFREMAP: UIF status bit remapping
  - 0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
  - 1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

- **Bit 10** Reserved, must be kept at reset value.

- **Bits 9:8** CKD[1:0]: Clock division
  - This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (tDTS) used by the dead-time generators and the digital filters (Tlx).
  - 00: tDTS=CK_INT
  - 01: tDTS=2*CK_INT
  - 10: tDTS=4*CK_INT
  - 11: Reserved, do not program this value

- **Bit 7** ARPE: Auto-reload preload enable
  - 0: TIMx_ARR register is not buffered
  - 1: TIMx_ARR register is buffered

- **Bits 6:4** Reserved, must be kept at reset value.

- **Bit 3** OPM: One pulse mode
  - 0: Counter is not stopped at update event
  - 1: Counter stops counting at the next update event (clearing the bit CEN)

- **Bit 2** URS: Update request source
  - This bit is set and cleared by software to select the UEV event sources.
  - 0: Any of the following events generate an update interrupt or DMA request if enabled.
    These events can be:
    - Counter overflow/underflow
    - Setting the UG bit
    - Update generation through the slave mode controller
  - 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.
**Bit 1 UDIS: Update disable**

This bit is set and cleared by software to enable/disable UEV event generation.

- **0**: UEV enabled. The Update (UEV) event is generated by one of the following events:
  - Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

- **1**: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

**Bit 0 CEN: Counter enable**

- **0**: Counter disabled
- **1**: Counter enabled

*Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

---

### 24.6.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

**Address offset: 0x04**

**Reset value: 0x0000**

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

*Bits 15:10 Reserved, must be kept at reset value.*

**Bit 9 OIS1N: Output Idle state 1 (OC1N output)**

- **0**: OC1N=0 after a dead-time when MOE=0
- **1**: OC1N=1 after a dead-time when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).*

**Bit 8 OIS1: Output Idle state 1 (OC1 output)**

- **0**: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
- **1**: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).*

*Bits 7:4 Reserved, must be kept at reset value.*

**Bit 3 CCDS: Capture/compare DMA selection**

- **0**: CCx DMA request sent when CCx event occurs
- **1**: CCx DMA requests sent when update event occurs

**Bit 2 CCUS: Capture/compare control update selection**

- **0**: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.
- **1**: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

*Note: This bit acts only on channels that have a complementary output.*

**Bit 1 Reserved, must be kept at reset value.**
### Bit 0 CCPC: Capture/compare preloaded control
- 0: CCxE, CCxNE and OCxM bits are not preloaded
- 1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.

*Note: This bit acts only on channels that have a complementary output.*

### 24.6.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

**Address offset: 0x0C**

**Reset value: 0x0000**

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

**Bits 15:10 Reserved, must be kept at reset value.**

**Bit 9 CC1DE: Capture/Compare 1 DMA request enable**
- 0: CC1 DMA request disabled
- 1: CC1 DMA request enabled

**Bit 8 UDE: Update DMA request enable**
- 0: Update DMA request disabled
- 1: Update DMA request enabled

**Bit 7 BIE: Break interrupt enable**
- 0: Break interrupt disabled
- 1: Break interrupt enabled

**Bit 6 Reserved, must be kept at reset value.**

**Bit 5 COMIE: COM interrupt enable**
- 0: COM interrupt disabled
- 1: COM interrupt enabled

**Bits 4:2 Reserved, must be kept at reset value.**

**Bit 1 CC1IE: Capture/Compare 1 interrupt enable**
- 0: CC1 interrupt disabled
- 1: CC1 interrupt enabled

**Bit 0 UIE: Update interrupt enable**
- 0: Update interrupt disabled
- 1: Update interrupt enabled
24.6.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10
Reset value: 0x0000

| Bit 15:10 Reserved, must be kept at reset value. |
| Bit 9 CC1OF: Capture/Compare 1 overcapture flag |
| This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'. |
| 0: No overcapture has been detected |
| 1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set |
| Bit 8 Reserved, must be kept at reset value. |
| Bit 7 BIF: Break interrupt flag |
| This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active. |
| 0: No break event occurred |
| 1: An active level has been detected on the break input |
| Bit 6 Reserved, must be kept at reset value. |
| Bit 5 COMIF: COM interrupt flag |
| This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software. |
| 0: No COM event occurred |
| 1: COM interrupt pending |
| Bits 4:2 Reserved, must be kept at reset value. |
| Bit 1 CC1IF: Capture/Compare 1 interrupt flag |
| **If channel CC1 is configured as output:** |
| This flag is set by hardware when the counter matches the compare value. It is cleared by software. |
| 0: No match. |
| 1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow |
| **If channel CC1 is configured as input:** |
| This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register. |
| 0: No input capture occurred |
| 1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity) |
24.6.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17)

Address offset: 0x14
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-8</td>
<td>Reserved, must be kept at reset value.</td>
<td>w</td>
</tr>
<tr>
<td>7</td>
<td>BG: Break generation</td>
<td>w</td>
</tr>
<tr>
<td>6</td>
<td>Reserved, must be kept at reset value.</td>
<td>w</td>
</tr>
<tr>
<td>5</td>
<td>COMG: Capture/Compare control update generation</td>
<td>w</td>
</tr>
<tr>
<td>4-2</td>
<td>Reserved, must be kept at reset value.</td>
<td>w</td>
</tr>
<tr>
<td>1</td>
<td>CC1G: Capture/Compare 1 generation</td>
<td>w</td>
</tr>
<tr>
<td>0</td>
<td>UG: Update generation</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG**: Capture/Compare control update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

*Note: This bit acts only on channels that have a complementary output.*

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A capture/compare event is generated on channel 1:

**If channel CC1 is configured as output:**
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

**If channel CC1 is configured as input:**
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).
### TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17)

Address offset: 0x18  
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Input capture mode:**

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

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:4  **IC1F[3:0]**: Input capture 1 filter
This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:
- 0000: No filter, sampling is done at fDTS
- 0001: fSAMPLING=fCK_INT, N=2
- 0010: fSAMPLING=fCK_INT, N=4
- 0011: fSAMPLING=fCK_INT, N=8
- 0100: fSAMPLING=fDTG/2, N=
- 0101: fSAMPLING=fDTG/4, N=8
- 0110: fSAMPLING=fDTG/4, N=6
- 0111: fSAMPLING=fDTG/4, N=8
- 1000: fSAMPLING=fDTG/8, N=6
- 1001: fSAMPLING=fDTG/8, N=8
- 1010: fSAMPLING=fDTG/16, N=5
- 1011: fSAMPLING=fDTG/16, N=6
- 1100: fSAMPLING=fDTG/16, N=8
- 1101: fSAMPLING=fDTG/32, N=5
- 1110: fSAMPLING=fDTG/32, N=6
- 1111: fSAMPLING=fDTG/32, N=8

Bits 3:2  **IC1PSC[1:0]**: Input capture 1 prescaler
This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
- 00: no prescaler, capture is done each time an edge is detected on the capture input.
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0  **CC1S[1:0]**: Capture/Compare 1 Selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, IC1 is mapped on TI1
- 10: CC1 channel is configured as input, IC1 is mapped on TI2
- 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER)*

### 24.6.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17)
Address offset: 0x18
Reset value: 0x0000 0000
The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Output compare mode:**
Bits 31:17 Reserved, must be kept at reset value.

Bits 15:7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.
0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
0100: Force inactive level - OC1REF is forced low.
0101: Force active level - OC1REF is forced high.
0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.
0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.
All other values: Reserved

**Note:** 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).
2: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

**Note:** 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).
2: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.
24.6.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20
Reset value: 0x0000

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 CC1NP: Capture/Compare 1 complementary output polarity
CC1 channel configured as output:
0: OC1N active high
1: OC1N active low
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to the description of CC1P.

Note: 1. This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).
2. On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.
Bit 2  **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1  **CC1P**: Capture/Compare 1 output polarity

**CC1 channel configured as output:**

0: OC1 active high
1: OC1 active low

**CC1 channel configured as input:**

The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

00: Non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).
01: Inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode).
10: Reserved, do not use this configuration.
1: Non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

**Note:** 1. This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).
   2. On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0  **CC1E**: Capture/Compare 1 output enable

**CC1 channel configured as output:**

0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

**CC1 channel configured as input:**

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.

0: Capture disabled
1: Capture enabled
Table 119. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)

<table>
<thead>
<tr>
<th>Control bits</th>
<th>Output states(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOE bit</td>
<td>OSSI bit</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

**Note:** The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and AFIO registers.

### 24.6.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

**Address offset:** 0x24

**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CNT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

774/1234 RM0444 Rev 2
Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

### 24.6.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28
Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency (CK_CNT) is equal to \( f_{CK_PSC} / (PSC[15:0] + 1) \).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in "reset mode").

### 24.6.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C
Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the Section 24.4.1: Time-base unit on page 703 for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
24.6.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30
Reset value: 0x0000

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

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]:** Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

24.6.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCR1[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:0 **CCR1[15:0]:** Capture/Compare 1 value

**If channel CC1 is configured as output:**

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

**If channel CC1 is configured as input:**

CCR1 is the counter value transferred by the last input capture 1 event (IC1).
24.6.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44
Reset value: 0x0000 0000

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSR, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 BKBID: Break Bidirectional
0: Break input BRK in input mode
1: Break input BRK in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 BKDSRM: Break Disarm
0: Break input BRK is armed
1: Break input BRK is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
Bits 25:20  Reserved, must be kept at reset value.

Bits 19:16  BKF[3:0]: Break filter

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, BRK acts asynchronously
0001: f_{SAMPLING}=f_{CK\_INT}, \ N=2
0010: f_{SAMPLING}=f_{CK\_INT}, \ N=4
0011: f_{SAMPLING}=f_{CK\_INT}, \ N=8
0100: f_{SAMPLING}=f_{DTS}/2, \ N=6
0101: f_{SAMPLING}=f_{DTS}/2, \ N=8
0110: f_{SAMPLING}=f_{DTS}/4, \ N=6
0111: f_{SAMPLING}=f_{DTS}/4, \ N=8
1000: f_{SAMPLING}=f_{DTS}/8, \ N=6
1001: f_{SAMPLING}=f_{DTS}/8, \ N=8
1010: f_{SAMPLING}=f_{DTS}/16, \ N=5
1011: f_{SAMPLING}=f_{DTS}/16, \ N=6
1100: f_{SAMPLING}=f_{DTS}/32, \ N=5
1101: f_{SAMPLING}=f_{DTS}/32, \ N=6
1110: f_{SAMPLING}=f_{DTS}/32, \ N=8
1111: f_{SAMPLING}=f_{DTS}/32, \ N=8

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15  MOE: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)

See OC/OCN enable description for more details (Section 24.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 772).

Bit 14  AOE: Automatic output enable

0: MOE can be set only by software
1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13  BKP: Break polarity

0: Break input BRK is active low
1: Break input BRK is active high

Note: 1. This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
2. Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12  BKE: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled
1: Break inputs (BRK and CCS clock failure event) enabled

Note: 1. This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
2. Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
Bit 11 OSSR: Off-state selection for Run mode
This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.
See OC/OCN enable description for more details (Section 24.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 772).
0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)
1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).
Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 OSSI: Off-state selection for Idle mode
This bit is used when MOE=0 on channels configured as outputs.
See OC/OCN enable description for more details (Section 24.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 772).
0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)
1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)
Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 LOCK[1:0]: Lock configuration
These bits offer a write protection against software errors.
00: LOCK OFF - No bit is write protected
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.
Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 DTG[7:0]: Dead-time generator setup
This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5]=0xx => DT=DTG[7:0]x tDTS
DTG[7:5]=10x => DT=(64+DTG[5:0])xtdtg with Ttdtg=2xtDTS
DTG[7:5]=110 => DT=(32+DTG[4:0])xtdtg with Ttdtg=8xDTS
DTG[7:5]=111 => DT=(32+DTG[4:0])xtdtg with Ttdtg=16xDTS
Example if T_DTS=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps
Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).
24.6.15  TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

Address offset: 0x48
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:13</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>12:8</td>
<td>DBL[4:0]: DMA burst length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00000: 1 transfer,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00001: 2 transfers,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00010: 3 transfers,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10001: 18 transfers.</td>
<td></td>
</tr>
<tr>
<td>7:5</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>4:0</td>
<td>DBA[4:0]: DMA base address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00000: TIMx_CR1,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00001: TIMx_CR2,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00010: TIMx_SMCR,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.</td>
<td></td>
</tr>
</tbody>
</table>

24.6.16  TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

Address offset: 0x4C
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>DMAB[15:0]: DMA register for burst accesses</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).</td>
<td></td>
</tr>
</tbody>
</table>
24.6.17 TIM16 alternate function register 1 (TIM16_AF1)

Address offset: 0x60
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-12</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>BKCMP2P: BRK COMP2 input polarity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: COMP2 input is active low</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: COMP2 input is active high</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>BKCMP1P: BRK COMP1 input polarity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: COMP1 input is active low</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: COMP1 input is active high</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>BKINP: BRK BKIN input polarity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: BKIN input is active low</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: BKIN input is active high</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</td>
<td></td>
</tr>
<tr>
<td>8-3</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>
Bit 2  **BKCMP2E**: BRK COMP2 enable
This bit enables the COMP2 for the timer’s BRK input. COMP2 output is ‘ORed’ with the other BRK sources.
- 0: COMP2 input disabled
- 1: COMP2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 1  **BKCMP1E**: BRK COMP1 enable
This bit enables the COMP1 for the timer’s BRK input. COMP1 output is ‘ORed’ with the other BRK sources.
- 0: COMP1 input disabled
- 1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 0  **BKINE**: BRK BKIN input enable
This bit enables the BKIN alternate function input for the timer’s BRK input. BKIN input is ‘ORed’ with the other BRK sources.
- 0: BKIN input disabled
- 1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

### 24.6.18  TIM16 alternate function register 1 (TIM16_AF1)

Address offset: 0x60
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-30</td>
<td>Res.</td>
<td>RW</td>
</tr>
<tr>
<td>29-28</td>
<td>Res.</td>
<td>RW</td>
</tr>
<tr>
<td>27-26</td>
<td>Res.</td>
<td>RW</td>
</tr>
<tr>
<td>25-24</td>
<td>Res.</td>
<td>RW</td>
</tr>
<tr>
<td>23-22</td>
<td>Res.</td>
<td>RW</td>
</tr>
<tr>
<td>21-20</td>
<td>Res.</td>
<td>RW</td>
</tr>
<tr>
<td>19-18</td>
<td>Res.</td>
<td>RW</td>
</tr>
<tr>
<td>17-16</td>
<td>BKCM P2E</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td>BKCM P1P</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td>BKINP</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td>BKCM P2E</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td>BKCM P1E</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td>BKINE</td>
<td>RW</td>
</tr>
</tbody>
</table>
Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity
This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP2 input is active low
1: COMP2 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 10 **BKCMP1P**: BRK COMP1 input polarity
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP1 input is active low
1: COMP1 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 9 **BKINP**: BRK BKIN input polarity
This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: BKIN input is active low
1: BKIN input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **BKCMP2E**: BRK COMP2 enable
This bit enables the COMP2 for the timer’s BRK input. COMP2 output is ‘ORed’ with the other BRK sources.
0: COMP2 input disabled
1: COMP2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 1 **BKCMP1E**: BRK COMP1 enable
This bit enables the COMP1 for the timer’s BRK input. COMP1 output is ‘ORed’ with the other BRK sources.
0: COMP1 input disabled
1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 0 **BKINE**: BRK BKIN input enable
This bit enables the BKIN alternate function input for the timer’s BRK input. BKIN input is ‘ORed’ with the other BRK sources.
0: BKIN input disabled
1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*
### 24.6.19 TIM16 input selection register (TIM16_TISEL)

Address offset: 0x68  
Reset value: 0x0000 0000

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

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

**Bits 3:0** **TISEL[3:0]** selects T1[0] to T1[15] input
- 0000: TIM16_CH1 input  
- 0001: LSI  
- 0010: LSE  
- 0011: RTC wakeup  
Others: Reserved

### 24.6.20 TIM17 alternate function register 1 (TIM17_AF1)

Address offset: 0x60  
Reset value: 0x0000 0001

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

Bits 31:12 Reserved, must be kept at reset value.

**Bit 11** **BKCM2P**: BRK COMP2 input polarity  
This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.  
0: COMP2 input is active low  
1: COMP2 input is active high  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

**Bit 10** **BKCM1P**: BRK COMP1 input polarity  
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.  
0: COMP1 input is active low  
1: COMP1 input is active high  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*
Bit 9 **BKINP**: BRK BKIN input polarity
This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: BKIN input is active low
1: BKIN input is active high
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **BKCMP2E**: BRK COMP2 enable
This bit enables the COMP2 for the timer’s BRK input. COMP2 output is ‘ORed’ with the other BRK sources.
0: COMP2 input disabled
1: COMP2 input enabled
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable
This bit enables the COMP1 for the timer’s BRK input. COMP1 output is ‘ORed’ with the other BRK sources.
0: COMP1 input disabled
1: COMP1 input enabled
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable
This bit enables the BKIN alternate function input for the timer’s BRK input. BKIN input is ‘ORed’ with the other BRK sources.
0: BKIN input disabled
1: BKIN input enabled
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

---

### 24.6.21 TIM17 input selection register (TIM17_TISEL)

Address offset: 0x68
Reset value: 0x0000 0000

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

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

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM17_CH1 input
0001: reserved
0010: HSE/32
0011: MCO
Others: Reserved
## 24.6.22 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | TIMx_CR1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Res. |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | TIMx_CR2      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Res. |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | TIMx_DIER     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Res. |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | TIMx_SR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Res. |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x14   | TIMx_EGR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Res. |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x18   | TIMx_CCMR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Output Compare mode | OC1M[3] | OC1M[2:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Res. |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x1C   | TIMx_CCMR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Input Capture mode | IC1F[3:0] | IC1F[3:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Res. |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x20   | TIMx_CCER     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | CC1NP | CC1NP |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x24   | TIMx_CNT      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | CNT[15:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x28   | TIMx_PSC      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | PSC[15:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2C   | TIMx_ARR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | ARR[15:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

### Table 120. TIM16/TIM17 register map and reset values

- **Offset**: The offset of the register in hexadecimal format.
- **Register name**: The name of the register.
- **Register**: The description of the register.
- **Reset value**: The reset value of the register in hexadecimal format.

The table provides a detailed view of the TIM16/TIM17 register map and their reset values.
### Table 120. TIM16/TIM17 register map and reset values (continued)

| Offset | Register name          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x30   | TIMx_RCR               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x34   | TIMx_CCR1              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x44   | TIMx_BDTR              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x48   | TIMx_DCR               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x4C   | TIMx_DMAR              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x60   | TIM16_AF1              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x60   | TIM17_AF1              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x68   | TIM16_TISEL            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x68   | TIM17_TISEL            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.2 on page 54 for the register boundary addresses.
25 Low-power timer (LPTIM)

25.1 Introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby and Shutdown mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. Also, the LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

25.2 LPTIM main features

- 16 bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: LSE, LSI, HSI16 or APB clock
  - External clock source over LPTIM input (working with no LP oscillator running, used by Pulse Counter application)
- 16 bit ARR autoreload register
- 16 bit compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable Digital Glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode

25.3 LPTIM implementation

Table 121 describes LPTIM implementation on STM32G0x1 devices: the full set of features is implemented in LPTIM1. LPTIM2 supports a smaller set of features, but is otherwise identical to LPTIM1.

<table>
<thead>
<tr>
<th>LPTIM modes/features(1)</th>
<th>LPTIM1</th>
<th>LPTIM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder mode</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

1. X = supported.
25.4 LPTIM functional description

25.4.1 LPTIM block diagram

Figure 271. Low-power timer block diagram (LPTIM1 and LPTIM2(1))

1. LPTIM2 has only the input channel 1, no input channel 2
2. lptim_out is the internal LPTIM output signal that can be connected to internal peripherals.

25.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 122. LPTIM input/output pins

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM_IN1</td>
<td>Digital input</td>
<td>LPTIM Input 1 from GPIO pin on mux input 0</td>
</tr>
<tr>
<td>LPTIM_IN2</td>
<td>Digital input</td>
<td>LPTIM Input 2 from GPIO pin on mux input 0</td>
</tr>
<tr>
<td>LPTIM_ETR</td>
<td>Digital input</td>
<td>LPTIM external trigger GPIO pin</td>
</tr>
<tr>
<td>LPTIM_OUT</td>
<td>Digital output</td>
<td>LPTIM Output GPIO pin</td>
</tr>
</tbody>
</table>

Table 123. LPTIM internal signals

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_pclk</td>
<td>Digital input</td>
<td>LPTIM APB clock domain</td>
</tr>
<tr>
<td>lptim_ker_ck</td>
<td>Digital input</td>
<td>LPTIM kernel clock</td>
</tr>
<tr>
<td>lptim_in1_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 1</td>
</tr>
</tbody>
</table>
25.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter:

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_in1_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 2</td>
</tr>
<tr>
<td>lptim_in1_mux3</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 3</td>
</tr>
<tr>
<td>lptim_in2_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input 2(1) connected to mux input 1</td>
</tr>
<tr>
<td>lptim_in2_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input 2(1) connected to mux input 2</td>
</tr>
<tr>
<td>lptim_in2_mux3</td>
<td>Digital input</td>
<td>Internal LPTIM input 2(1) connected to mux input 3</td>
</tr>
<tr>
<td>lptim_ext_trigx</td>
<td>Digital input</td>
<td>LPTIM external trigger input x</td>
</tr>
<tr>
<td>lptim_out</td>
<td>Digital output</td>
<td>LPTIM counter output</td>
</tr>
<tr>
<td>lptim_it</td>
<td>Digital output</td>
<td>LPTIM global interrupt</td>
</tr>
<tr>
<td>lptim_wakeup</td>
<td>Digital output</td>
<td>LPTIM wakeup event</td>
</tr>
</tbody>
</table>

1. Only applies to LPTIM1

Table 123. LPTIM internal signals (continued)

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_ext_trig0</td>
<td>GPIO pin as LPTIM1_ETR alternate function</td>
<td></td>
</tr>
<tr>
<td>lptim_ext_trig1</td>
<td>RTC ALARM A</td>
<td></td>
</tr>
<tr>
<td>lptim_ext_trig2</td>
<td>RTC ALARM B</td>
<td></td>
</tr>
<tr>
<td>lptim_ext_trig3</td>
<td>TAMPO input detection</td>
<td></td>
</tr>
<tr>
<td>lptim_ext_trig4</td>
<td>TAMPO input detection</td>
<td></td>
</tr>
<tr>
<td>lptim_ext_trig5</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>lptim_ext_trig6</td>
<td>COMP1_OUT</td>
<td></td>
</tr>
<tr>
<td>lptim_ext_trig7</td>
<td>COMP2_OUT</td>
<td></td>
</tr>
</tbody>
</table>

Table 124. LPTIM1 external trigger connection

<table>
<thead>
<tr>
<th>TRIGSEL</th>
<th>External trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_ext_trig0</td>
<td>GPIO pin as LPTIM1_ETR alternate function</td>
</tr>
<tr>
<td>lptim_ext_trig1</td>
<td>RTC ALARM A</td>
</tr>
<tr>
<td>lptim_ext_trig2</td>
<td>RTC ALARM B</td>
</tr>
<tr>
<td>lptim_ext_trig3</td>
<td>TAMPO input detection</td>
</tr>
<tr>
<td>lptim_ext_trig4</td>
<td>TAMPO input detection</td>
</tr>
<tr>
<td>lptim_ext_trig5</td>
<td>Reserved</td>
</tr>
<tr>
<td>lptim_ext_trig6</td>
<td>COMP1_OUT</td>
</tr>
<tr>
<td>lptim_ext_trig7</td>
<td>COMP2_OUT</td>
</tr>
</tbody>
</table>

Table 125. LPTIM2 external trigger connection

<table>
<thead>
<tr>
<th>TRIGSEL</th>
<th>External trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_ext_trig0</td>
<td>GPIO pin as LPTIM2_ETR alternate function</td>
</tr>
<tr>
<td>lptim_ext_trig1</td>
<td>RTC ALARM A</td>
</tr>
<tr>
<td>lptim_ext_trig2</td>
<td>RTC ALARM B</td>
</tr>
<tr>
<td>lptim_ext_trig3</td>
<td>TAMPO input detection</td>
</tr>
<tr>
<td>lptim_ext_trig4</td>
<td>TAMPO input detection</td>
</tr>
<tr>
<td>lptim_ext_trig5</td>
<td>Reserved</td>
</tr>
<tr>
<td>lptim_ext_trig6</td>
<td>COMP1_OUT</td>
</tr>
<tr>
<td>lptim_ext_trig7</td>
<td>COMP2_OUT</td>
</tr>
</tbody>
</table>
The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be chosen among APB, LSI, LSE or HSI16 sources through the Reset and Clock controller (RCC). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM may run in one of these two possible configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM either from APB or any other embedded oscillator including LSE, LSI and HSI16.
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or Pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM will use an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal should also be provided (first configuration). In this case, the internal clock signal frequency should be at least four times higher than the external clock signal frequency.
25.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals, such as embedded comparators), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source should first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

The digital filters are divided into two groups:

- The first group of digital filters protects the LPTIM external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that should be detected on one of the LPTIM inputs to consider a signal level change as a valid transition. Figure 272 shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

![Figure 272. Glitch filter timing diagram](image)

Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT and TRGFLT bits to ‘0’. In that case, an external analog filter may be used to protect the LPTIM external inputs against glitches.

25.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] 3-bit field. The table below lists all the possible division ratios:

<table>
<thead>
<tr>
<th>programming</th>
<th>dividing factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>/1</td>
</tr>
<tr>
<td>001</td>
<td>/2</td>
</tr>
<tr>
<td>010</td>
<td>/4</td>
</tr>
</tbody>
</table>

Table 129. Prescaler division ratios
25.4.7 Trigger multiplexer

The LPTIM counter may be started either by software or after the detection of an active edge on one of the 8 trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals ‘00’, The LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than ‘00’, TRIGSEL[2:0] is used to select which of the 8 trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. So after a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it will be ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled will be discarded by hardware.

25.4.8 Operating mode

The LPTIM features two operating modes:

- The Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when reaching the ARR value.

One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event will re-start the timer. Any trigger event occurring after the counter starts and before the counter reaches ARR will be discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the counter register has stopped (contains zero value), will start the counter for a new one-shot counting cycle as shown in Figure 273.
- Set-once mode activated:

It should be noted that when the WAVE bit-field in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in Figure 274.

In case of software start (TRIGEN[1:0] = '00'), the SNGSTRT setting will start the counter for one-shot counting.

**Continuous mode**

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set will start the counter for continuous counting. Any subsequent external trigger event will be discarded as shown in Figure 275.

In case of software start (TRIGEN[1:0] = '00'), setting CNTSTRT will start the counter for continuous counting.
SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (The ENABLE bit is set to ‘1’). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT will switch the LPTIM to the One-shot mode. The counter (if active) will stop as soon as it reaches ARR.

If the One-shot mode was previously selected, setting CNTSTRT will switch the LPTIM to the Continuous mode. The counter (if active) will restart as soon as it reaches ARR.

### 25.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event will start the timer, any successive trigger event will reset the counter and the timer will restart.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

### 25.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CMP (compare register), are used to generate several different waveforms on LPTIM output.

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CMP. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT registers.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset.
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require that the LPTIM_ARR register value be strictly greater than the LPTIM_CMP register value.
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to ‘0’ forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to ‘1’ forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value will change immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated. Figure 276 below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL bit.

**Figure 276. Waveform generation**

![Diagram showing three waveforms: PWM, One shot, Set once, with Pol = 0 and Pol = 1](MS32467v2)

### 25.4.11 Register update

The LPTIM_ARR register and LPTIM_CMP register are updated immediately after the APB bus write operation, or at the end of the current period if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR and the LPTIM_CMP registers are updated:

- When the PRELOAD bit is reset to ‘0’, the LPTIM_ARR and the LPTIM_CMP registers are immediately updated after any write access.
- When the PRELOAD bit is set to ‘1’, the LPTIM_ARR and the LPTIM_CMP registers are updated at the end of the current period, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the
counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag and the CMPOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register and the LPTIM_CMP register.

After a write to the LPTIM_ARR register or the LPTIM_CMP register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag or the CMPOK flag be set, will lead to unpredictable results.

25.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM Input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source will be used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- **CKSEL = 0**: the LPTIM is clocked by an internal clock source
  - **COUNTMODE = 0**: The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - **COUNTMODE = 1**: The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM. Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal should never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).

- **CKSEL = 1**: the LPTIM is clocked by an external clock source
  
  **COUNTMODE value is don’t care.**

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external Input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

  Since the signal injected on the LPTIM external Input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

25.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock is needed before the LPTIM is actually enabled.
The LPTIM_CFGR and LPTIM_IER registers must be modified only when the LPTIM is disabled.

25.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register to zero, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bit-field to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic will elapse before the reset is taken into account. This will make the LPTIM counter count few extra pluses between the time when the reset is trigger and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.

- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register will reset its content to zero. Asynchronous reset should be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset should be applied only when there is enough insurance that no toggle will occur on the LPTIM Input1.

It should be noted that to read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. So developer should make sure that these two mechanisms are used exclusively.

25.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.
Direction change is signalized by the two Down and Up flags in the LPTIM_ISR register. Also, an interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to ‘1’. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder’s position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

<table>
<thead>
<tr>
<th>Active edge</th>
<th>Level on opposite signal (Input1 for Input2, Input2 for Input1)</th>
<th>Input1 signal</th>
<th>Input2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rising</td>
<td>Falling</td>
<td>Rising</td>
</tr>
<tr>
<td>Rising Edge</td>
<td>High</td>
<td>Down</td>
<td>No count</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
<td>Up</td>
<td>No count</td>
</tr>
<tr>
<td>Falling Edge</td>
<td>High</td>
<td>No count</td>
<td>Up</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
<td>No count</td>
<td>Down</td>
</tr>
<tr>
<td>Both Edges</td>
<td>High</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
<td>Up</td>
<td>Down</td>
</tr>
</tbody>
</table>

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to ‘0’. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be ‘000’).
25.4.16 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the DBG_LPTIM_STOP configuration bit in the DBG module.

25.5 LPTIM low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. LPTIM interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Low-power run</td>
<td>No effect.</td>
</tr>
<tr>
<td>Low-power sleep</td>
<td>No effect. LPTIM interrupts cause the device to exit the Low-power sleep mode.</td>
</tr>
<tr>
<td>Stop 0 / Stop 1</td>
<td>No effect when LPTIM is clocked by LSE or LSI. LPTIM interrupts cause the device to exit Stop 0 and Stop 1.</td>
</tr>
<tr>
<td>Standby</td>
<td>The LPTIM peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.</td>
</tr>
</tbody>
</table>
25.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_IER register:
- Compare match
- Auto-reload match (whatever the direction if encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (encoder mode), programmable (up / down / both).

Note: If any bit in the LPTIM_IER register (Interrupt Enable Register) is set after that its corresponding flag in the LPTIM_ISR register (Status Register) is set, the interrupt is not asserted.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compare match</td>
<td>Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the compare register (LPTIM_CMP).</td>
</tr>
<tr>
<td>Auto-reload match</td>
<td>Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the Auto-reload register (LPTIM_ARR).</td>
</tr>
<tr>
<td>External trigger event</td>
<td>Interrupt flag is raised when an external trigger event is detected</td>
</tr>
<tr>
<td>Auto-reload register update OK</td>
<td>Interrupt flag is raised when the write operation to the LPTIM_ARR register is complete.</td>
</tr>
<tr>
<td>Compare register update OK</td>
<td>Interrupt flag is raised when the write operation to the LPTIM_CMP register is complete.</td>
</tr>
</tbody>
</table>
| Direction change | Used in Encoder mode. Two interrupt flags are embedded to signal direction change:  
– UP flag signals up-counting direction change  
– DOWN flag signals down-counting direction change. |

25.7 LPTIM registers

25.7.1 LPTIM interrupt and status register (LPTIM_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

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

RM0444 Rev 2 801/1234
Bit 31:7  Reserved, must be kept at reset value.

Bit 6 **DOWN**: Counter direction change up to down
In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

*Note*: *If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 25.3: LPTIM implementation.*

Bit 5 **UP**: Counter direction change down to up
In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

*Note*: *If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 25.3: LPTIM implementation.*

Bit 4 **ARROK**: Autoreload register update OK
ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMPOK**: Compare register update OK
CMPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_CMP register has been successfully completed.

Bit 2 **EXTTRIG**: External trigger edge event
EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match
ARRM is set by hardware to inform application that LPTIM_CNT register’s value reached the LPTIM_ARR register’s value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CMPM**: Compare match
The CMPM bit is set by hardware to inform application that LPTIM_CNT register value reached the LPTIM_CMP register’s value.

### 25.7.2 LPTIM interrupt clear register (LPTIM_ICR)
Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DOWN CF</td>
<td>UPCF</td>
<td>ARRO KCF</td>
<td>CMPO KCF</td>
<td>EXTTRIG CF</td>
<td>ARRM CF</td>
<td>CMPM CF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>
Bits 31:9 Reserved, must be kept at reset value.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clears the DOWN flag in the LPTIM_ISR register.

Note: *If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 25.3: LPTIM implementation.*

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clears the UP flag in the LPTIM_ISR register.

Note: *If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 25.3: LPTIM implementation.*

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMPOKCF**: Compare register update OK clear flag
Writing 1 to this bit clears the CMPOK flag in the LPTIM_ISR register

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CMPMCF**: Compare match clear flag
Writing 1 to this bit clears the CMP flag in the LPTIM_ISR register

### 25.7.3 LPTIM interrupt enable register (LPTIM_IER)

Address offset: 0x008
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable

0: DOWN interrupt disabled
1: DOWN interrupt enabled

Note: *If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 25.3: LPTIM implementation.*
Caution: The LPTIM_IER register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’)

25.7.4 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENC</td>
<td>COUNT MODE</td>
<td>PRELOAD</td>
<td>WAVPOL</td>
<td>WAVE</td>
<td>TIMOUT</td>
<td>TRIGEN[1:0]</td>
<td>ENC</td>
<td>COUNT MODE</td>
<td>PRELOAD</td>
<td>WAVPOL</td>
<td>WAVE</td>
<td>TIMOUT</td>
<td>TRIGEN[1:0]</td>
<td>ENC</td>
<td>COUNT MODE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:30 Reserved, must be kept at reset value.
Bit 29 Reserved, must be kept at reset value.
Bits 28:25 Reserved, must be kept at reset value.

Bit 24 ENC: Encoder mode enable
The ENC bit controls the Encoder mode
0: Encoder mode disabled
1: Encoder mode enabled

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 25.3: LPTIM implementation.
Bit 23 **COUNTMODE**: counter mode enabled
The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:
0: the counter is incremented following each internal clock pulse
1: the counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode
The PRELOAD bit controls the LPTIM_ARR and the LPTIM_CMP registers update modality
0: Registers are updated after each APB bus write access
1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity
The WAVPOL bit controls the output polarity
0: The LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CMP registers
1: The LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CMP registers

Bit 20 **WAVE**: Waveform shape
The WAVE bit controls the output shape
0: Deactivate Set-once mode, PWM or One Pulse waveform depending on how the timer was started, CNTSTRT for PWM or SNGSTRT for One Pulse waveform.
1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable
The TIMOUT bit controls the Timeout feature
0: A trigger event arriving when the timer is already started will be ignored
1: A trigger event arriving when the timer is already started will reset and restart the counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity
The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:
00: software trigger (counting start is initiated by software)
01: rising edge is the active edge
10: falling edge is the active edge
11: both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector
The TRIGSEL bits select the trigger source that will serve as a trigger event for the LPTIM among the below 8 available sources:
000: lptim_ext_trig0
001: lptim_ext_trig1
010: lptim_ext_trig2
011: lptim_ext_trig3
100: lptim_ext_trig4
101: lptim_ext_trig5
110: lptim_ext_trig6
111: lptim_ext_trig7
See Section 25.4.3: LPTIM input and trigger mapping for details.

Bit 12 Reserved, must be kept at reset value.
Bits 11:9  **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

- 000: /1
- 001: /2
- 010: /4
- 011: /8
- 100: /16
- 101: /32
- 110: /64
- 111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6  **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature.

- 00: any trigger active level change is considered as a valid trigger
- 01: trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
- 10: trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
- 11: trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.
Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’).

25.7.5 LPTIM control register (LPTIM_CR)

Address offset: 0x010
Reset value: 0x0000 0000
25.7.6 LPTIM compare register (LPTIM_CMP)

Address offset: 0x014
Reset value: 0x0000 0000

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

CMP[15:0]

808/1234 RM0444 Rev 2
Caution: The LPTIM_CMP register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

25.7.7 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018  
Reset value: 0x0000 0001

| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0 | ARR[15:0]: Auto reload value |
| ARR[15:0] | rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw |
| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0 | ARR[15:0]: Auto reload value |

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

25.7.8 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C  
Reset value: 0x0000 0000

| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0 | CNT[15:0]: Counter value |

Caution: The LPTIM_CNT register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

When the LPTIM is running with an asynchronous clock, reading the LPTIM_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two returned values are identical. It should be noted that for a reliable LPTIM_CNT register read access, two consecutive read accesses must be performed and compared. A read access can be considered reliable when the values of the two consecutive read accesses are equal.
25.7.9  LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024
Reset value: 0x0000 0000

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

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

Bits 5:4  **IN2SEL[1:0]: LPTIM input 2 selection**

- The IN2SEL bits control the LPTIM Input 2 multiplexer, which connect LPTIM Input 2 to one of the available inputs.
  - 00: lptim_in2_mux0
  - 01: lptim_in2_mux1
  - 10: lptim_in2_mux2
  - 11: lptim_in2_mux3

For connection details refer to Section 25.4.3: LPTIM input and trigger mapping.

Note: If the LPTIM does not support encoder mode feature, these bits are reserved. Please refer to Section 25.3: LPTIM implementation.

Bits 3:2  Reserved, must be kept at reset value.

Bits 1:0  **IN1SEL[1:0]: LPTIM input 1 selection**

- The IN1SEL bits control the LPTIM Input 1 multiplexer, which connects LPTIM Input 1 to one of the available inputs.
  - 00: lptim_in1_mux0
  - 01: lptim_in1_mux1
  - 10: lptim_in1_mux2
  - 11: lptim_in1_mux3

For connection details refer to Section 25.4.3: LPTIM input and trigger mapping.

**Caution:** The LPTIM_CFGR2 register must only be modified when the LPTIM is disabled (ENABLE bit reset to '0').
### 25.7.10 LPTIM register map

The following table summarizes the LPTIM registers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bit 0</th>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Bit 4</th>
<th>Bit 5</th>
<th>Bit 6</th>
<th>Bit 7</th>
<th>Bit 8</th>
<th>Bit 9</th>
<th>Bit 10</th>
<th>Bit 11</th>
<th>Bit 12</th>
<th>Bit 13</th>
<th>Bit 14</th>
<th>Bit 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>LPTIM_ISR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x004</td>
<td>LPTIM_ICR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x008</td>
<td>LPTIM_IER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x00C</td>
<td>LPTIM_CFGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ENC</td>
<td>COUNTMODE</td>
<td>PRELOAD</td>
<td>WAVEPOL</td>
<td>WAVE</td>
<td>TIMOUT</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x010</td>
<td>LPTIM_CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x014</td>
<td>LPTIM_CMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CMP[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x018</td>
<td>LPTIM_ARR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ARR[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0x01C</td>
<td>LPTIM_CNT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CNT[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x020</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x024</td>
<td>LPTIM_CFGR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IN2SEL[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
1. If LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 25.3: LPTIM implementation.

Refer to Section 2.2 on page 54 for the register boundary addresses.
26 Infrared interface (IRTIM)

An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with USART1, USART4 (on STM32G071xx and STM32G081xx) or USART2 (STM32G031xx and STM32G041xx), TIM16 and TIM17 as shown in Figure 278.

To generate the infrared remote control signals, the IR interface must be enabled and TIM16 channel 1 (TIM16_OC1) and TIM17 channel 1 (TIM17_OC1) must be properly configured to generate correct waveforms.

The infrared receiver can be implemented easily through a basic input capture mode.

Figure 278. IRTIM internal hardware connections

TIM17_CH1
TIM16_CH1
USART1
USART4
IR_MOD[1:0]
IR_POL
IR_OUT

3. USART4 on STM32G071xx and STM32G081xx, and USART2 on STM32G031xx and STM32G041xx.

All standard IR pulse modulation modes can be obtained by programming the two timer output compare channels.

TIM17 is used to generate the high frequency carrier signal, while TIM16 or alternatively USART1 or USART4 generates the modulation envelope according to the setting of the IR_MOD[1:0] bits in the SYSCFG_CFRG1 register.

The polarity of the output signal from IRTIM is controlled by the IR_POL bit in the SYSCFG_CFRG1 register and could be inverted by setting of this bit.

The infrared function is output on the IR_OUT pin. The activation of this function is done through the GPIOx_AFRx register by enabling the related alternate function bit.

The high sink LED driver capability (only available on the PB9 pin) can be activated through the I2C_PB9_FMP bit in the SYSCFG_CFRG1 register and used to sink the high current needed to directly control an infrared LED.
27 Independent watchdog (IWDG)

27.1 Introduction

The devices feature an embedded watchdog peripheral that offers a combination of high safety level, timing accuracy and flexibility of use. The Independent watchdog peripheral detects and solves malfunctions due to software failure, and triggers system reset when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. For further information on the window watchdog, refer to Section 28 on page 823.

27.2 IWDG main features

- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional Reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

27.3 IWDG functional description

27.3.1 IWDG block diagram

*Figure 279* shows the functional blocks of the independent watchdog module.

![Figure 279. Independent watchdog block diagram](image)

1. The register interface is located in the VDD voltage domain. The watchdog function is located in the VDD voltage domain, still functional in Stop and Standby modes.
When the independent watchdog is started by writing the value 0x0000 CCCC in the **IWDG key register (IWDG_KR)**, the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0x0000 AAAA is written in the **IWDG key register (IWDG_KR)**, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

Once running, the IWDG cannot be stopped.

### 27.3.2 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the **IWDG window register (IWDG_WINR)**.

If the reload operation is performed while the counter is greater than the value stored in the **IWDG window register (IWDG_WINR)**, then a reset is provided.

The default value of the **IWDG window register (IWDG_WINR)** is 0x0000 0FFF, so if it is not updated, the window option is disabled.

As soon as the window value is changed, a reload operation is performed in order to reset the downcounter to the **IWDG reload register (IWDG_RLR)** value and ease the cycle number calculation to generate the next reload.

#### Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the **IWDG key register (IWDG_KR)**.
2. Enable register access by writing 0x0000 5555 in the **IWDG key register (IWDG_KR)**.
3. Write the IWDG prescaler by programming **IWDG prescaler register (IWDG_PR)** from 0 to 7.
4. Write the **IWDG reload register (IWDG_RLR)**.
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Write to the **IWDG window register (IWDG_WINR)**. This automatically refreshes the counter value in the **IWDG reload register (IWDG_RLR)**.

**Note:** Writing the window value allows to refresh the Counter value by the RLR when **IWDG status register (IWDG_SR)** is set to 0x0000 0000.

#### Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the **IWDG key register (IWDG_KR)**.
2. Enable register access by writing 0x0000 5555 in the **IWDG key register (IWDG_KR)**.
3. Write the prescaler by programming the **IWDG prescaler register (IWDG_PR)** from 0 to 7.
4. Write the **IWDG reload register (IWDG_RLR)**.
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Refresh the counter value with IWDG_RLR (IWDG_KR = 0x0000 AAAA).
27.3.3 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and generates a reset unless the IWDG key register (IWDG_KR) is written by the software before the counter reaches end of count or if the downcounter is reloaded inside the window.

27.3.4 Register access protection

Write access to IWDG prescaler register (IWDG_PR), IWDG reload register (IWDG_RLR) and IWDG window register (IWDG_WINR) is protected. To modify them, the user must first write the code 0x0000 5555 in the IWDG key register (IWDG_KR). A write access to this register with a different value will break the sequence and register access will be protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is on going.

27.3.5 Debug mode

When the device enters Debug mode (core halted), the IWDG counter either continues to work normally or stops, depending on the configuration of the corresponding bit in DBGMCU freeze register.
27.4  IWDG registers

Refer to Section 1.2 on page 50 for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

27.4.1  IWDG key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by Standby mode)

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

<table>
<thead>
<tr>
<th></th>
<th>KEY[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>w</td>
</tr>
<tr>
<td></td>
<td>w</td>
</tr>
<tr>
<td></td>
<td>w</td>
</tr>
<tr>
<td></td>
<td>w</td>
</tr>
<tr>
<td></td>
<td>w</td>
</tr>
<tr>
<td></td>
<td>w</td>
</tr>
<tr>
<td></td>
<td>w</td>
</tr>
<tr>
<td></td>
<td>w</td>
</tr>
<tr>
<td></td>
<td>w</td>
</tr>
<tr>
<td></td>
<td>w</td>
</tr>
<tr>
<td></td>
<td>w</td>
</tr>
<tr>
<td></td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  KEY[15:0]: Key value (write only, read 0x0000)

These bits must be written by software at regular intervals with the key value 0xAAAA, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 0x5555 to enable access to the IWDG_PR, IWDG_RLR and IWDG_WINR registers (see Section 27.3.4: Register access protection)

Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)
27.4.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04
Reset value: 0x0000 0000

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

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

Bits 2:0 **PR[2:0]:** Prescaler divider

These bits are write access protected see *Section 27.3.4: Register access protection.* They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (IWDG_SR)* must be reset in order to be able to change the prescaler divider.

- 000: divider /4
- 001: divider /8
- 010: divider /16
- 011: divider /32
- 100: divider /64
- 101: divider /128
- 110: divider /256
- 111: divider /256

**Note:** Reading this register returns the prescaler value from the V_DD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the *IWDG status register (IWDG_SR)* is reset.
27.4.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08
Reset value: 0x0000 0FFF (reset by Standby mode)

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected see *Register access protection.* They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *IWDG key register (IWDG_KR).* The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to the datasheet for the timeout information.

The RVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

**Note:** *Reading this register returns the reload value from the V<sub>DD</sub> voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on it. For this reason the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.*
### 27.4.4 IWDG status register (IWDG_SR)

Address offset: 0x0C
Reset value: 0x0000 0000 (not reset by Standby mode)

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

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

- **Bit 2 W Vũ**: Watchdog counter window value update
  
  This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to five LSI cycles).
  
  Window value can be updated only when WVU bit is reset.
  
  This bit is generated only if generic "window" = 1

- **Bit 1 R Vũ**: Watchdog counter reload value update
  
  This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to five LSI cycles).
  
  Reload value can be updated only when RVU bit is reset.

- **Bit 0 P Vũ**: Watchdog prescaler value update
  
  This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain (takes up to five LSI cycles).
  
  Prescaler value can be updated only when PVU bit is reset.

**Note:** If several reload, prescaler, or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, and to wait until WVU bit is reset before changing the window value. However, after updating the prescaler and/or the reload/window value it is not necessary to wait until RVU or PVU or WVU is reset before continuing code execution except in case of low-power mode entry.
### 27.4.5  IWDG window register (IWDG_WINR)

Address offset: 0x10  
Reset value: 0x0000 0FFF (reset by Standby mode)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>13</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>12</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>6</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>4</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>3</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>2</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>1</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

**WIN[11:0]:** Watchdog counter window value  
These bits are write access protected, see Section 27.3.4, they contain the high limit of the window value to be compared with the downcounter.  
To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x0  
The WVU bit in the **IWDG status register (IWDG_SR)** must be reset in order to be able to change the reload value.  
**Note:** Reading this register returns the reload value from the VDD voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the **IWDG status register (IWDG_SR)** is reset.
### 27.4.6 IWDG register map

The following table gives the IWDG register map and reset values.

Table 134. IWDG register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>IWDG_KR</td>
<td>0x04</td>
<td>IWDG_PR</td>
<td>0x08</td>
<td>IWDG_RLR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x04</td>
<td></td>
<td>0x0C</td>
<td>IWDG_SR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x10</td>
<td></td>
<td></td>
<td>IWDG_WINR</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
</tbody>
</table>

Refer to *Section 2.2 on page 54* for the register boundary addresses.
28 System window watchdog (WWDG)

28.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications which require the watchdog to react within an accurate timing window.

28.2 WWDG main features

- Programmable free-running downcounter
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x40
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window (see Figure 281)
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40.

28.3 WWDG functional description

If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit downcounter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to Figure 280 for the WWDG block diagram.
28.3.1 WWDG block diagram

Figure 280. Watchdog block diagram

28.3.2 Enabling the watchdog

When the user option WWDG_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

When the user option WWDG_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

28.3.3 Controlling the downcounter

This downcounter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG_CR register (see Figure 281). The Configuration register (WWDG_CFR) contains the high limit of the window: To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x3F. Figure 281 describes the window watchdog process.

Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

28.3.4 Advanced watchdog interrupt feature

The early wakeup interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by setting the EWI bit in the WWDG_CFR register. When the downcounter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR)
can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case, the corresponding interrupt service routine (ISR) should reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

Note: When the EWI interrupt cannot be served, e.g. due to a system lock in a higher priority task, the WWDG reset is eventually generated.

28.3.5 How to program the watchdog timeout

Use the formula in Figure 281 to calculate the WWDG timeout.

---

**Warning:** When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

---

**Figure 281. Window watchdog timing diagram**

![Window watchdog timing diagram](MS47266V1)
The formula to calculate the timeout value is given by:

\[ t_{\text{WWDG}} = t_{\text{PCLK}} \times 4096 \times 2^{\text{WDGTB}[1:0]} \times (T[5:0] + 1) \quad (\text{ms}) \]

where:
- \( t_{\text{WWDG}} \): WWDG timeout
- \( t_{\text{PCLK}} \): APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, let's assume APB frequency is equal to 48 MHz, WDGTB[1:0] is set to 3 and \( T[5:0] \) is set to 63:

\[ t_{\text{WWDG}} = \left( \frac{1}{48000} \right) \times 4096 \times 2^3 \times (63 + 1) = 43.69 \text{ms} \]

Refer to the datasheet for the minimum and maximum values of the \( t_{\text{WWDG}} \).

### 28.3.6 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details refer to Section 37: Debug support (DBG).

### 28.4 WWDG registers

Refer to Section 1.2 on page 50 for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

### 28.4.1 Control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

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

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

<table>
<thead>
<tr>
<th>wrist</th>
<th>rs</th>
<th>r</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
</table>

WDGA    T[6:0]
28.4.2 **Configuration register (WWDG_CFR)**

**Address offset:** 0x004  
**Reset value:** 0x0000 007F

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
| Bit 7 WDGA: Activation bit  
This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.  
0: Watchdog disabled  
1: Watchdog enabled |
| Bit 6:0 T[6:0]: 7-bit counter (MSB to LSB)  
These bits contain the value of the watchdog counter, decremented every \( (4096 \times 2^{WDGTB[1:0]}) \) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared). |

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
</table>
| Bit 13:11 WDGTB[2:0]: Timer base  
The timebase of the prescaler can be modified as follows:  
000: CK Counter Clock (PCLK div 4096) div 1  
001: CK Counter Clock (PCLK div 4096) div 2  
010: CK Counter Clock (PCLK div 4096) div 4  
011: CK Counter Clock (PCLK div 4096) div 8  
100: CK Counter Clock (PCLK div 4096) div 16  
101: CK Counter Clock (PCLK div 4096) div 32  
110: CK Counter Clock (PCLK div 4096) div 64  
111: CK Counter Clock (PCLK div 4096) div 128 |
| Bit 9 EWI: Early wakeup interrupt  
When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset. |
| Bits 6:0 W[6:0]: 7-bit window value  
These bits contain the window value to be compared with the downcounter. |
28.4.3 Status register (WWDG_SR)

Address offset: 0x008
Reset value: 0x0000 0000

| Bit 31:1 | Reserved, must be kept at reset value. |
| Bit 0   | EWIF: Early wakeup interrupt flag |

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing ‘0’. Writing ‘1’ has no effect. This bit is also set if the interrupt is not enabled.

28.4.4 WWDG register map

The following table gives the WWDG register map and reset values.

| Offset  | Register  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000   | WWDG_CR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x004   | WWDG_CFR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008   | WWDG_SR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.2 on page 54 for the register boundary addresses.
29 Real-time clock (RTC)

29.1 Introduction
The RTC provides an automatic wakeup to manage all low-power modes.
The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.
As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).
The RTC is functional in $V_{BAT}$ mode.

29.2 RTC main features
The RTC supports the following features (see Figure 282: RTC block diagram):
• Calendar with subsecond, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
• Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
• Two programmable alarms.
• On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
• Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
• Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
• Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to $V_{BAT}$ mode.
• 17-bit auto-reload wakeup timer (WUT) for periodic events with programmable resolution and period.
The RTC is supplied through a switch that takes power either from the $V_{DD}$ supply when present or from the $V_{BAT}$ pin.
The RTC clock sources can be:
• A 32.768 kHz external crystal (LSE)
• An external resonator or oscillator (LSE)
• The internal low power RC oscillator (LSI, with typical frequency of 32 kHz)
• The high-speed external clock (HSE), divided by a prescaler in the RCC.
The RTC is functional in $V_{BAT}$ mode and in all low-power modes when it is clocked by the LSE. When clocked by the LSI, the RTC is not functional in $V_{BAT}$ mode, but is functional in all low-power modes except Shutdown mode.
All RTC events (Alarm, WakeUp Timer, Timestamp) can generate an interrupt and wakeup the device from the low-power modes.
29.3 RTC functional description

29.3.1 RTC block diagram

Figure 282. RTC block diagram
29.3.2 RTC pins and internal signals

Table 136. RTC input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_TS</td>
<td>Input</td>
<td>RTC timestamp input</td>
</tr>
<tr>
<td>RTC_REFIN</td>
<td>Input</td>
<td>RTC 50 or 60 Hz reference clock input</td>
</tr>
<tr>
<td>RTC_OUT1</td>
<td>Output</td>
<td>RTC output 1</td>
</tr>
<tr>
<td>RTC_OUT2</td>
<td>Output</td>
<td>RTC output 2</td>
</tr>
</tbody>
</table>

- RTC_OUT1 and RTC_OUT2 which selects one of the following two outputs:
  - CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
  - TAMPALRM: This output is the OR between TAMP and ALARM outputs.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wakeup outputs. TAMP is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 137. RTC internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtc_ker_ck</td>
<td>Input</td>
<td>RTC kernel clock, also named RTCCLK in this document</td>
</tr>
<tr>
<td>rtc_pclk</td>
<td>Input</td>
<td>RTC APB clock</td>
</tr>
<tr>
<td>rtc_its</td>
<td>Input</td>
<td>RTC internal timestamp event</td>
</tr>
<tr>
<td>rtc_tamp_evt</td>
<td>Input</td>
<td>Tamper event (internal or external) detected in TAMP peripheral</td>
</tr>
<tr>
<td>rtc_it</td>
<td>Output</td>
<td>RTC interrupts (refer to Section 29.5: RTC interrupts for details)</td>
</tr>
<tr>
<td>rtc_alra_trg</td>
<td>Output</td>
<td>RTC alarm A event detection trigger</td>
</tr>
<tr>
<td>rtc_alrb_trg</td>
<td>Output</td>
<td>RTC alarm B event detection trigger</td>
</tr>
<tr>
<td>rtc_wut_trg</td>
<td>Output</td>
<td>RTC wakeup timer event detection trigger</td>
</tr>
<tr>
<td>rtc_calovf</td>
<td>Output</td>
<td>RTC calendar overflow</td>
</tr>
</tbody>
</table>

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or $V_{BAT}$ when the selected clock is not LSE. Refer to Section 29.4: RTC low-power modes for more details.
The triggers outputs can be used as triggers for other peripherals.

29.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the Battery Backup Domain (V_{BAT}) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

Both RTC and TAMP peripherals provide functions on these I/Os (refer to Section 30: Tamper and backup registers (TAMP)).

RTC.\_OUT1, RTC.\_TS and TAMP.\_IN1 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes and in V_{BAT} mode.

The output mechanism follows the priority order shown in Table 139.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtc_its</td>
<td>From power controller (PWR): main power loss/snow switch to V_{BAT} detection output</td>
</tr>
<tr>
<td>rtc_tamp_evt</td>
<td>From TAMP peripheral: tamp_evt</td>
</tr>
<tr>
<td>rtc_calovf</td>
<td>To TAMP peripheral: tamp_itamp5</td>
</tr>
</tbody>
</table>

---

### Table 138. RTC interconnection

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtc_its</td>
<td>From power controller (PWR): main power loss/snow switch to V_{BAT} detection output</td>
</tr>
<tr>
<td>rtc_tamp_evt</td>
<td>From TAMP peripheral: tamp_evt</td>
</tr>
<tr>
<td>rtc_calovf</td>
<td>To TAMP peripheral: tamp_itamp5</td>
</tr>
</tbody>
</table>

---

### Table 139. PC13 configuration(1)

<table>
<thead>
<tr>
<th>PC13 Pin function</th>
<th>OSEL[1:0] (ALARM output enable)</th>
<th>TAMPOE (TAMPER output enable)</th>
<th>COE (CALIB output enable)</th>
<th>OUT2EN</th>
<th>TAMPA_TYPE</th>
<th>TAMPA_PU</th>
<th>TAMPA1E (TAMP_IN1 input enable)</th>
<th>TSE (RTC_TS input enable)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>TAMPALRM output</td>
<td>00</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
</tbody>
</table>
### Table 139. PC13 configuration (1) (continued)

<table>
<thead>
<tr>
<th>PC13 Pin function</th>
<th>OSEL[1:0] (ALARM output enable)</th>
<th>TAMPOE (TAMPER output enable)</th>
<th>COE (CALIB output enable)</th>
<th>OUT2EN</th>
<th>TAMPALRM_TYPE</th>
<th>TAMPA_PU</th>
<th>TAMP1E (TAMP_IN1 input enable)</th>
<th>TSE (RTC_TS input enable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No pull</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don't care</td>
<td>Don't care 1</td>
<td>0</td>
<td>Don't care</td>
<td>Don't care</td>
<td>Don't care</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>1</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>1</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>1</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>Internal pull-up</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>1</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>1</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>CALIB output PP</td>
<td>00</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>TAMP_IN1 input floating</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>0</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC_TS and TAMP_IN1 input floating</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>0</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC_TS input floating</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>0</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In addition, it is possible to output RTC_OUT2 on PA4 pin thanks to OUT2EN bit. This output is not available in VBAT mode. The different functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, COE and OUT2EN configuration, as show in table Table 140.

For PA4, the GPIO should be configured as an alternate function.

Table 140. RTC_OUT mapping

<table>
<thead>
<tr>
<th>OSEL[1:0] bits (ALARM output enable)</th>
<th>COE bit (CALIB output enable)</th>
<th>OUT2EN bit</th>
<th>RTC_OUT1 on PC13</th>
<th>RTC_OUT2 on PA4</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>0</td>
<td>CALIB</td>
<td>-</td>
</tr>
<tr>
<td>01 or 10 or 11</td>
<td>Don’t care</td>
<td>TAMPALRM</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>CALIB</td>
</tr>
<tr>
<td>01 or 10 or 11</td>
<td>0</td>
<td>TAMPALRM</td>
<td>-</td>
<td>TAMPALRM</td>
</tr>
<tr>
<td>01 or 10 or 11</td>
<td>1</td>
<td>TAMPALRM</td>
<td>CALIB</td>
<td></td>
</tr>
</tbody>
</table>

29.3.4 Clock and prescalers

The RTC clock source (RTCCLOCK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to Section 5: Reset and clock control (RCC).
A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see Figure 282: RTC block diagram):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is $2^{22}$. This corresponds to a maximum input frequency of around 4 MHz.

The ck_apre clock is used to clock the binary RTC_SSR subseconds downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see Section 29.3.7: Periodic auto-wakeup for details).

### 29.3.5 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see Section 29.6.10: RTC shift control register (RTC_SHIFTR)). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the

\[
\frac{f_{\text{CK\_APRE}}}{f_{\text{RTCCLK}}} = \frac{f_{\text{RTCCLK}}}{\text{PREDIV\_A} + 1}
\]

The ck_apre clock is used to clock the binary RTC_SSR subseconds downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

\[
\frac{f_{\text{CK\_SPRE}}}{f_{\text{RTCCLK}}} = \frac{f_{\text{RTCCLK}}}{(\text{PREDIV\_S} + 1) \times (\text{PREDIV\_A} + 1)}
\]

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see Section 29.3.7: Periodic auto-wakeup for details).
BYPASHAD control bit in the RTC CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC SSR, RTC TR or RTC DR registers in BYPASHAD = 0 mode, the frequency of the APB clock \( f_{APB} \) must be at least 7 times the frequency of the RTC clock \( f_{RTCCLK} \).

The shadow registers are reset by system reset.

29.3.6 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

The alarm interrupt is enabled through the ALRAIE bit in the RTC CR register.

Caution:

If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC PRER register must be at least 3 to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC CR register.

29.3.7 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC CR register.

The wakeup timer clock input \( ck\_wut \) can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.
  
  When RTCCLK is LSE (32.768 kHz), this allows to configure the wakeup interrupt period from 122 \( \mu \)s to 32 s, with a resolution down to 61 \( \mu \)s.

- \( ck\_spre \) (usually 1 Hz internal clock)
  
  When \( ck\_spre \) frequency is 1 Hz, this allows to achieve a wakeup time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case \( 2^{16} \) is added to the 16-bit counter current value. When the initialization sequence is complete (see Programming the wakeup timer on page 838), the timer starts counting down. When the wakeup function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in
the RTC_SR register, and the wakeup counter is automatically reloaded with its reload value (RTC_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wakeup flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop and Standby) have no influence on the wakeup timer.

29.3.8 RTC initialization and configuration

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD = 0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable RTC registers write access.

After RTC domain reset, some of the RTC registers are write-protected.

Writing to the protected RTC registers is enabled by writing a key into the Write Protection register, RTC_WPR.

The following steps are required to unlock the write protection on the protected RTC registers.
1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:
1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.

2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).

3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register.

4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.

5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

**Note:** After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its RTC domain reset default value (0x00).

To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

### Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

### Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

**Note:** Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

### Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. This step must be skipped in calendar initialization mode. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wakeup auto-reload value WUT[15:0], and the wakeup clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again.
The wakeup timer restarts down-counting. The WUTWF bit is cleared up to 2 RTCCLK clocks cycles after WUTE is cleared, due to clock synchronization.

29.3.9 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB1 clock frequency (f_PCLK) must be equal to or greater than seven times the RTC clock frequency (f_RTCCLK). This ensures a secure behavior of the synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycles. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wakeup and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to Calendar initialization and configuration on page 837): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to Section 29.3.11: RTC synchronization): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.
Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

29.3.10 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a RTC domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wakeup timer register (RTC_WUTR), and the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the RTC domain reset one (refer to RCC for details about RTC clock sources not affected by system reset). When a RTC domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

29.3.11 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

RTC_SSR contains the value of the synchronous prescaler counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]). The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of 1 / (PREDIV_S + 1) seconds. The shift operation consists of adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this will delay the clock. If at the same time the ADD1S bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this will advance the clock.

Caution: Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow will occur.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.
Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

29.3.12 RTC reference clock detection

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:
- PREDIV_A = 0x007F
- PREDIV_S = 0x00FF

Note: RTC_REFIN clock detection is not available in Standby mode.

29.3.13 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about $2^{20}$ RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, cal_cnt[19:0], clocked by RTCCLK.
The smooth calibration register (RTC_CALR) specifies the number of RTCCLK clock cycles to be masked during the 32-second cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked.
- Setting CALM[2] to 1 causes four additional cycles to be masked.
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of RTCCLK pulses to be masked during the 32-second cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra RTCCLK pulse every 211 RTCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the 32-second cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (FcAL) given the input frequency (FRTCCLK) is as follows:

\[ F_{CAL} = F_{RTCCLK} \times \left[1 + \frac{(CALP \times 512 - CALM)}{(2^{20} + CALM - CALP \times 512)}\right] \]

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV_A less than 3, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every 32 seconds. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each 32-second cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

\[ F_{CAL} = F_{RTCCLK} \times \left[1 + \frac{(256 - CALM)}{(2^{20} + CALM - 256)}\right] \]

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.
Verifying the RTC calibration

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16-second calibration cycle period.

In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:
1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

29.3.14 Timestamp function

Timestamp is enabled by setting the TSE or ITSE bits of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a tamper event is detected on the TAMP_INx pin.

When ITSE is set:
The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the VBAT supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

**Note:** TSF is set 2 ck_apre cycles after the timestamp event occurs due to synchronization process.

There is no delay in the setting of TSOVF. This means that if two timestamp events are close together, TSOVF can be seen as ‘1’ while TSF is still ‘0’. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

**Caution:** If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

Optionally, a tamper event can cause a timestamp to be recorded. See the description of the TAMPTS control bit in the RTC control register (RTC_CR).

### 29.3.15 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is fRTCCLK/64. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is fRTCCLK/(256 * (PREDIV_A+1)). This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

**Note:** When the CALIB output is selected, the RTC_OUT1 pin is automatically configured but the RTC_OUT2 pin must be set as alternate function.

When COSEL is cleared, the CALIB output is the output of the 6th stage of the asynchronous prescaler.

When COSEL is set, the CALIB output is the output of the 8th stage of the synchronous prescaler.

### 29.3.16 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.
When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wakeup flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

**TAMPALRM output**

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

*Note:* Once the TAMPALRM output is enabled, it has priority over CALIB on RTC_OUT1.

When TAMPALRM output is selected, the RTC_OUT1 pin is automatically configured but the RTC_OUT2 pin must be set as alternate function. In case the TAMPALRM is configured open-drain in the RTC, the RTC_OUT1 GPIO must be configured as input.

### 29.4 RTC low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. RTC interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.</td>
</tr>
<tr>
<td>Shutdown</td>
<td>The RTC remains active when the RTC clock source is LSE. RTC interrupts cause the device to exit the Shutdown mode.</td>
</tr>
</tbody>
</table>

The table below summarizes the RTC pins and functions capability in all modes.

<table>
<thead>
<tr>
<th>Functions</th>
<th>Functional in all low-power modes except Standby and Shutdown modes</th>
<th>Functional in Standby and Shutdown mode</th>
<th>Functional in V&lt;sub&gt;BAT&lt;/sub&gt; mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_TS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RTC_REFIN</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RTC_OUT1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RTC_OUT2</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
29.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register. The interrupt output is also activated.

Table 143. Interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag(1)</th>
<th>Enable control bit(2)</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop and Standby mode</th>
<th>Exit from Shutdown mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC</td>
<td>Alarm A</td>
<td>ALRAF</td>
<td>ALRAIE</td>
<td>write 1 in CALRAF</td>
<td>Yes</td>
<td>Yes(3)</td>
<td>Yes(4)</td>
</tr>
<tr>
<td></td>
<td>Alarm B</td>
<td>ALRBF</td>
<td>ALRBIE</td>
<td>write 1 in CALRBF</td>
<td>Yes</td>
<td>Yes(3)</td>
<td>Yes(4)</td>
</tr>
<tr>
<td></td>
<td>Timestamp</td>
<td>TSF</td>
<td>TSIE</td>
<td>write 1 in CTSF</td>
<td>Yes</td>
<td>Yes(3)</td>
<td>Yes(4)</td>
</tr>
<tr>
<td></td>
<td>Wakeup timer interrupt</td>
<td>WUTF</td>
<td>WUTIE</td>
<td>write 1 in CWUTF</td>
<td>Yes</td>
<td>Yes(3)</td>
<td>Yes(4)</td>
</tr>
</tbody>
</table>

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. Wakeup from Stop and Standby modes is possible only when the RTC clock source is LSE or LSI.
4. Wakeup from Shutdown modes is possible only when the RTC clock source is LSE.

29.6 RTC registers

Refer to Section 1.2 on page 50 of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

29.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to Calendar initialization and configuration on page 837 and Reading the calendar on page 839.

This register is write protected. The write access procedure is described in RTC register write protection on page 837.

Address offset: 0x00

RTC domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)
The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to Calendar initialization and configuration on page 837 and Reading the calendar on page 839.

This register is write protected. The write access procedure is described in RTC register write protection on page 837.

Address offset: 0x04

RTC domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

<table>
<thead>
<tr>
<th>Address</th>
<th>Bits description</th>
<th>Format</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:23</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 22</td>
<td><strong>PM</strong>: AM/PM notation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0: AM or 24-hour format</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: PM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 21:20</td>
<td><strong>HT[1:0]</strong>: Hour tens in BCD format</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 19:16</td>
<td><strong>HU[3:0]</strong>: Hour units in BCD format</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 15</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 14:12</td>
<td><strong>MNT[2:0]</strong>: Minute tens in BCD format</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 11:8</td>
<td><strong>MNU[3:0]</strong>: Minute units in BCD format</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 7</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 6:4</td>
<td><strong>ST[2:0]</strong>: Second tens in BCD format</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 3:0</td>
<td><strong>SU[3:0]</strong>: Second units in BCD format</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**29.6.2 RTC date register (RTC_DR)**

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to Calendar initialization and configuration on page 837 and Reading the calendar on page 839.

This register is write protected. The write access procedure is described in RTC register write protection on page 837.

Address offset: 0x04

RTC domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

<table>
<thead>
<tr>
<th>Address</th>
<th>Bits description</th>
<th>Format</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:24</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 23:20</td>
<td><strong>YT[3:0]</strong>: Year tens in BCD format</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 19:16</td>
<td><strong>YU[3:0]</strong>: Year units in BCD format</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 15:13</td>
<td><strong>WDU[2:0]</strong>: Week day units</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000: forbidden</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001: Monday</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>111: Sunday</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 12</td>
<td><strong>MT</strong>: Month tens in BCD format</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 11:8</td>
<td><strong>MU[3:0]</strong>: Month units in BCD format</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
29.6.3 RTC sub second register (RTC_SSR)

Address offset: 0x08

RTC domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 DT[1:0]: Date tens in BCD format

Bits 3:0 DU[3:0]: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can’t roll over.

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

29.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in RTC register write protection on page 837.

Address offset: 0x0C

RTC domain reset value: 0x0000 0007

System reset value: not affected (except INIT, INITF, and RSF bits which are cleared to 0)

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 SS[15:0]: Sub second value

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

Note: SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

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

848/1234  RM0444 Rev 2
Bits 31:17  Reserved, must be kept at reset value.

Bit 16  **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Re-calibration on-the-fly*.

Bits 15:8  Reserved, must be kept at reset value.

Bit 7  **INIT**: Initialization mode

0: Free running mode
1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Bit 6  **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed
1: Calendar registers update is allowed

Bit 5  **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSRx, RTC_TRx and RTC_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized
1: Calendar shadow registers synchronized

Bit 4  **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (RTC domain reset state).

0: Calendar has not been initialized
1: Calendar has been initialized

Bit 3  **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending
1: A shift operation is pending
Bit 2 **WUTWF**: Wakeup timer write flag
   This bit is set by hardware when WUT value can be changed, after the WUTE bit has been
   set to 0 in RTC_CR.
   It is cleared by hardware in initialization mode.
   0: Wakeup timer configuration update not allowed except in initialization mode
   1: Wakeup timer configuration update allowed

Bit 1 **ALRBWF**: Alarm B write flag
   This bit is set by hardware when alarm B values can be changed, after the ALRBE bit has
   been set to 0 in RTC_CR.
   It is cleared by hardware in initialization mode.
   0: Alarm B update not allowed
   1: Alarm B update allowed

Bit 0 **ALRAWF**: Alarm A write flag
   This bit is set by hardware when alarm A values can be changed, after the ALRAE bit has
   been set to 0 in RTC_CR.
   It is cleared by hardware in initialization mode.
   0: Alarm A update not allowed
   1: Alarm A update allowed

### 29.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed
in two separate write accesses. Refer to *Calendar initialization and configuration on page 837*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 837*.

Address offset: 0x10

RTC domain reset value: 0x007F 00FF

System reset: not affected

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

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor
   This is the asynchronous division factor:
   \[ \text{ck}_{\text{apre}} \text{ frequency} = \frac{\text{RTCCLK} \text{ frequency}}{(PREDIV_A+1)} \]

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor
   This is the synchronous division factor:
   \[ \text{ck}_{\text{spre}} \text{ frequency} = \frac{\text{ck}_{\text{apre}} \text{ frequency}}{(PREDIV_S+1)} \]
29.6.6 RTC wakeup timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register is write protected. The write access procedure is described in RTC register write protection on page 837.

Address offset: 0x14
RTC domain reset value: 0x0000 FFFF
System reset: not affected

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

WUT[15:0]

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **WUT[15:0]:** Wakeup auto-reload value bits

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.

When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 1) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

29.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 837.

Address offset: 0x18
RTC domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT2 EN</td>
<td>TAMPR ALRM TYPE</td>
<td>TAMPR ALRM _PU</td>
<td>Res.</td>
<td>Res.</td>
<td>TAMP OE</td>
<td>TAMP TS</td>
<td>ITSE</td>
<td>COE</td>
<td>OSEL[1:0]</td>
<td>POL</td>
<td>COSEL</td>
<td>BKP</td>
<td>SUB1H</td>
<td>ADD1H</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

| TSIE | WUTIE | ALRBE IE | ALRA IE | TSE | WUTE | ALRBE | ALRAE | Res. | FMT | BYP SHAD | REFCK ON | TS EDGE | WUCKSEL[2:0] | rw |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
Bit 31 **OUT2EN**: RTC_OUT2 output enable
Setting this bit allows to remap the RTC outputs on RTC_OUT2 as follows:

- **OUT2EN = 0**: RTC output 2 disable
  - If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1
  - If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1

- **OUT2EN = 1**: RTC output 2 enable
  - If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2
  - If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2
  - If (OSEL ≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type
- 0: TAMPALRM is push-pull output
- 1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM_PU**: TAMPALRM pull-up enable
- 0: No pull-up is applied on TAMPALRM output
- 1: A pull-up is applied on TAMPALRM output

Bits 28:27 Reserved, must be kept at reset value.

Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM
- 0: The tamper flag is not routed on TAMPALRM
- 1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event
- 0: Tamper detection event does not cause a RTC timestamp to be saved
- 1: Save RTC timestamp on tamper detection event
  TAMPPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set after the tamper flags, therefore if TAMPTS and TSIE are set, it is recommended to disable the tamper interrupts in order to avoid servicing 2 interrupts.

Bit 24 **ITSE**: timestamp on internal event enable
- 0: Internal event timestamp disabled
- 1: Internal event timestamp enabled

Bit 23 **COE**: Calibration output enable
This bit enables the CALIB output
- 0: Calibration output disabled
- 1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection
These bits are used to select the flag to be routed to TAMPALRM output.
- 00: Output disabled
- 01: Alarm A output enabled
- 10: Alarm B output enabled
- 11: Wakeup output enabled

Bit 20 **POL**: Output polarity
This bit is used to configure the polarity of TAMPALRM output.
- 0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).
- 1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).
Bit 19 **COSEL**: Calibration output selection
When COE = 1, this bit selects which signal is output on CALIB.
0: Calibration output is 512 Hz
1: Calibration output is 1 Hz
These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to Section 29.3.15: Calibration clock output.

Bit 18 **BKP**: Backup
This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)
When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
Setting this bit has no effect when current hour is 0.
0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.
0: No effect
1: Adds 1 hour to the current time. This can be used for summer time change.

Bit 15 **TSIE**: Timestamp interrupt enable
0: Timestamp interrupt disable
1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wakeup timer interrupt enable
0: Wakeup timer interrupt disabled
1: Wakeup timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable
0: Alarm B interrupt disable
1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable
0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable
0: timestamp disable
1: timestamp enable

Bit 10 **WUTE**: Wakeup timer enable
0: Wakeup timer disabled
1: Wakeup timer enabled

*Note: When the wakeup timer is disabled, wait for WUTWF=1 before enabling it again.*

Bit 9 **ALRBE**: Alarm B enable
0: Alarm B disabled
1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable
0: Alarm A disabled
1: Alarm A enabled

Bit 7 Reserved, must be kept at reset value.
Real-time clock (RTC)  

Bit 6  **FMT**: Hour format  
0: 24 hour/day format  
1: AM/PM hour format  

Bit 5  **BYPHAD**: Bypass the shadow registers  
0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.  
1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.  

*Note: If the frequency of the APB1 clock is less than seven times the frequency of RTCCLK, BYPHAD must be set to 1.*  

Bit 4  **REFCONE**: RTC_REFIN reference clock detection enable (50 or 60 Hz)  
0: RTC_REFIN detection disabled  
1: RTC_REFIN detection enabled  

*Note: PREDIV_S must be 0x00FF.*  

Bit 3  **TSEDGE**: Timestamp event active edge  
0: RTC_TS input rising edge generates a timestamp event  
1: RTC_TS input falling edge generates a timestamp event  
TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.  

Bits 2:0  **WUCKSEL[2:0]**: ck_wut wakeup clock selection  
000: RTC/16 clock is selected  
001: RTC/8 clock is selected  
010: RTC/4 clock is selected  
011: RTC/2 clock is selected  
10x: ck_spre (usually 1 Hz) clock is selected  
11x: ck_spre (usually 1 Hz) clock is selected and $2^{16}$ is added to the WUT counter value  

*Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).*  

WUT = wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1] = 11.  

*Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.*  

*It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.*  

*ADD1H and SUB1H changes are effective in the next second.*  

### 29.6.8 RTC write protection register (RTC_WPR)  

Address offset: 0x24  
Reset value: 0x0000 0000  

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

*KEY[7:0]*  

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

Bits 7:0  **KEY[7:0]**: Write protection key
This byte is written by software.
Reading this byte always returns 0x00.
Refer to **RTC register write protection** for a description of how to unlock RTC register write protection.

### 29.6.9 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in **RTC register write protection on page 837**.

Address offset: 0x28

RTC domain reset value: 0x0000 0000

System reset: not affected

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

<table>
<thead>
<tr>
<th>CALP</th>
<th>CALW8</th>
<th>CALW 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

**Bit 15  CALP**: Increase frequency of RTC by 488.5 ppm
0: No RTCCCLK pulses are added.
1: One RTCCCLK pulse is effectively inserted every $2^{11}$ pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. If the input frequency is 32768 Hz, the number of RTCCCLK pulses added during a 32-second window is calculated as follows: $(512 \times \text{CALP}) - \text{CALM}$.

Refer to **Section 29.3.13: RTC smooth digital calibration**.

**Bit 14  CALW8**: Use an 8-second calibration cycle period
When CALW8 is set to 1, the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to **Section 29.3.13: RTC smooth digital calibration**.
Bit 13 **CALW16**: Use a 16-second calibration cycle period
When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.

*Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 29.3.13: RTC smooth digital calibration.*

Bits 12:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of $2^{20}$ RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See Section 29.3.13: RTC smooth digital calibration on page 841.

### 29.6.10 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in RTC register write protection on page 837.

Address offset: 0x2C

RTC domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD1S</td>
<td>Ref</td>
<td>Ref</td>
<td>Ref</td>
<td>Ref</td>
<td>Ref</td>
<td>Ref</td>
<td>Ref</td>
<td>Ref</td>
<td>Ref</td>
<td>Ref</td>
<td>Ref</td>
<td>Ref</td>
<td>Ref</td>
<td>Ref</td>
<td>Ref</td>
</tr>
<tr>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SUBFS[14:0]</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

- **ADD1S**: Add one second
  0: No effect
  1: Add one second to the clock/calendar
  This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).
  This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

- **Bits 30:15**: Reserved, must be kept at reset value.

- **SUBFS[14:0]**: Subtract a fraction of a second
  These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).
  The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:
  \[
  \text{Delay (seconds)} = \text{SUBFS} / (\text{PREDIV}_S + 1)
  \]
  A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:
  \[
  \text{Advance (seconds)} = (1 - (\text{SUBFS} / (\text{PREDIV}_S + 1)))
  \]
  *Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.*
29.6.11 **RTC timestamp time register (RTC_TSTR)**

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

Address offset: 0x30

RTC domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 PM: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 HT[1:0]: Hour tens in BCD format.

Bits 19:16 Hu[3:0]: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 MNT[2:0]: Minute tens in BCD format.

Bits 11:8 MNU[3:0]: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 ST[2:0]: Second tens in BCD format.

Bits 3:0 SU[3:0]: Second units in BCD format.

29.6.12 **RTC timestamp date register (RTC_TSDR)**

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

Address offset: 0x34

RTC domain reset value: 0x0000 0000

System reset: not affected

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

|r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  |
Bits 31:16  Reserved, must be kept at reset value.

Bits 15:13  **WDU[2:0]**: Week day units

Bit 12  **MT**: Month tens in BCD format

Bits 11:8  **MU[3:0]**: Month units in BCD format

Bits 7:6  Reserved, must be kept at reset value.

Bits 5:4  **DT[1:0]**: Date tens in BCD format

Bits 3:0  **DU[3:0]**: Date units in BCD format

### 29.6.13  RTC timestamp sub second register (RTC_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.

Address offset: 0x38

RTC domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **SS[15:0]**: Sub second value

**SS[15:0]** is the value of the synchronous prescaler counter when the timestamp event occurred.
29.6.14  RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in RTC register write protection on page 837.

Address offset: 0x40

RTC domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
| Bit 31  MSK4: Alarm A date mask
  0: Alarm A set if the date/day match
  1: Date/day don’t care in alarm A comparison

Bit 30  WDSEL: Week day selection
  0: DU[3:0] represents the date units
  1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28  DT[1:0]: Date tens in BCD format

Bits 27:24  DU[3:0]: Date units or day in BCD format

Bit 23  MSK3: Alarm A hours mask
  0: Alarm A set if the hours match
  1: Hours don’t care in alarm A comparison

Bit 22  PM: AM/PM notation
  0: AM or 24-hour format
  1: PM

Bits 21:20  HT[1:0]: Hour tens in BCD format

Bits 19:16  HU[3:0]: Hour units in BCD format

Bit 15  MSK2: Alarm A minutes mask
  0: Alarm A set if the minutes match
  1: Minutes don’t care in alarm A comparison

Bits 14:12  MNT[2:0]: Minute tens in BCD format

Bits 11:8  MNU[3:0]: Minute units in BCD format

Bit 7  MSK1: Alarm A seconds mask
  0: Alarm A set if the seconds match
  1: Seconds don’t care in alarm A comparison

Bits 6:4  ST[2:0]: Second tens in BCD format.

Bits 3:0  SU[3:0]: Second units in BCD format.
29.6.15 RTC alarm A sub second register (RTC_ALRMASSR)

This register can be written only when ALRAWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in RTC register write protection on page 837.

Address offset: 0x44

RTC domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>MASKSS[3:0]</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 27:24 MASKSS[3:0]: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
1: SS[14:1] are don’t care in alarm A comparison. Only SS[0] is compared.
2: SS[14:2] are don’t care in alarm A comparison. Only SS[1:0] are compared.
...
15: All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Note: The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 SS[14:0]: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.
29.6.16  RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in RTC register write protection on page 837.

Address offset: 0x48

RTC domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>MSK4</td>
<td>Alarm B date mask</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Alarm B set if the date and day match</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Date and day don’t care in alarm B comparison</td>
</tr>
<tr>
<td>30</td>
<td>WDSEL</td>
<td>Week day selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: DU[3:0] represents the date units</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: DU[3:0] represents the week day. DT[1:0] is don’t care.</td>
</tr>
<tr>
<td>29:28</td>
<td>DT[1:0]</td>
<td>Date tens in BCD format</td>
</tr>
<tr>
<td>27:24</td>
<td>DU[3:0]</td>
<td>Date units or day in BCD format</td>
</tr>
<tr>
<td>23</td>
<td>MSK3</td>
<td>Alarm B hours mask</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Alarm B set if the hours match</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Hours don’t care in alarm B comparison</td>
</tr>
<tr>
<td>22</td>
<td>PM</td>
<td>AM/PM notation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: AM or 24-hour format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: PM</td>
</tr>
<tr>
<td>21:20</td>
<td>HT[1:0]</td>
<td>Hour tens in BCD format</td>
</tr>
<tr>
<td>19:16</td>
<td>HU[3:0]</td>
<td>Hour units in BCD format</td>
</tr>
<tr>
<td>15</td>
<td>MSK2</td>
<td>Alarm B minutes mask</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Alarm B set if the minutes match</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Minutes don’t care in alarm B comparison</td>
</tr>
<tr>
<td>14:12</td>
<td>MNT[2:0]</td>
<td>Minute tens in BCD format</td>
</tr>
<tr>
<td>11:8</td>
<td>MNU[3:0]</td>
<td>Minute units in BCD format</td>
</tr>
<tr>
<td>7</td>
<td>MSK1</td>
<td>Alarm B seconds mask</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Alarm B set if the seconds match</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Seconds don’t care in alarm B comparison</td>
</tr>
<tr>
<td>6:4</td>
<td>ST[2:0]</td>
<td>Second tens in BCD format</td>
</tr>
<tr>
<td>3:0</td>
<td>SU[3:0]</td>
<td>Second units in BCD format</td>
</tr>
</tbody>
</table>
29.6.17  RTC alarm B sub second register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in Section: RTC register write protection.

Address offset: 0x4C

RTC domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27-24</td>
<td>MASKSS[3:0]: Mask the most-significant bits starting at this bit</td>
</tr>
<tr>
<td>23-15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14-0</td>
<td>SS[14:0]: Sub seconds value</td>
</tr>
</tbody>
</table>

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

Bits 27:24  MASKSS[3:0]: Mask the most-significant bits starting at this bit

- 0x0: No comparison on sub seconds for alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
- 0x1: SS[14:1] are don’t care in alarm B comparison. Only SS[0] is compared.
- 0x2: SS[14:2] are don’t care in alarm B comparison. Only SS[1:0] are compared.
- 0x3: SS[14:3] are don’t care in alarm B comparison. Only SS[2:0] are compared.
- ... 0x9: SS[14:12] are don’t care in alarm B comparison. SS[11:0] are compared.
- 0xA: SS[14:13] are don’t care in alarm B comparison. SS[12:0] are compared.
- 0xB: SS[14] is don’t care in alarm B comparison. SS[13:0] are compared.
- 0xC: All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15  Reserved, must be kept at reset value.

Bits 14:0  SS[14:0]: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

29.6.18  RTC status register (RTC_SR)

Address offset: 0x50

RTC domain reset value: 0x0000 0000

System reset: not affected
### RTC masked interrupt status register (RTC_MISR)

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

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

- **Bit 5 ITSF**: Internal timestamp flag
  - This flag is set by hardware when a timestamp on the internal event occurs.

- **Bit 4 TSOVF**: Timestamp overflow flag
  - This flag is set by hardware when a timestamp event occurs while TSF is already set.
  - It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

- **Bit 3 TSF**: Timestamp flag
  - This flag is set by hardware when a timestamp event occurs.
  - If ITSF flag is set, TSF must be cleared together with ITSF.

- **Bit 2 WUTF**: Wakeup timer flag
  - This flag is set by hardware when the wakeup auto-reload counter reaches 0.
  - This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

- **Bit 1 ALRBF**: Alarm B flag
  - This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).

- **Bit 0 ALRAF**: Alarm A flag
  - This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

**Note:** The bits of this register are cleared 2 APB clock cycles after setting their corresponding clear bit in the RTC_SCR register.

### 29.6.19 RTC masked interrupt status register (RTC_MISR)

- **Address offset:** 0x54
- **RTC domain reset value:** 0x0000 0000
- **System reset:** not affected
Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **ITSMF**: Internal timestamp masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestampinterrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow masked flag
This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp masked flag
This flag is set by hardware when a timestamp interrupt occurs. If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wakeup timer masked flag
This flag is set by hardware when the wakeup timer interrupt occurs. This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B masked flag
This flag is set by hardware when the alarm B interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag
This flag is set by hardware when the alarm A interrupt occurs.

### 29.6.20 RTC status clear register (RTC_SCR)

Address offset: 0x5C

RTC domain reset value: 0x0000 0000

System reset: not affected

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

Bit 5 **CITSF**: Clear internal timestamp flag
Writing 1 in this bit clears the ITSF bit in the RTC_SR register.

Bit 4 **CTSOVF**: Clear timestamp overflow flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register. If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.
Bit 2 **CWUTF**: Clear wakeup timer flag
   Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 **CALRBF**: Clear alarm B flag
   Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0 **CALRAF**: Clear alarm A flag
   Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.
# 29.6.21 RTC register map

## Table 144. RTC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset Register</th>
<th>Reset value</th>
<th>Offset Register</th>
<th>Offset Register</th>
<th>Offset Register</th>
<th>Offset Register</th>
<th>Offset Register</th>
<th>Offset Register</th>
<th>Offset Register</th>
<th>Offset Register</th>
<th>Offset Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>RTC_TR</td>
<td>PM</td>
<td>HT [1:0]</td>
<td>HU[3:0]</td>
<td>MNT[2:0]</td>
<td>MNU[3:0]</td>
<td>ST[2:0]</td>
<td>SU[3:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>RTC_DR</td>
<td>YT[3:0]</td>
<td>YU[3:0]</td>
<td>WDU[2:0]</td>
<td>MT</td>
<td>MU[3:0]</td>
<td>DT [1:0]</td>
<td>DU[3:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>RTC_SS</td>
<td>SS[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>RTC_ICS</td>
<td>RECAPF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>RTC_PRER</td>
<td>PREDIV_A[6:0]</td>
<td>PREDIV_S[14:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>RTC_WUTR</td>
<td>WUT[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>RTC_CR</td>
<td>WU[15:0]</td>
<td>BT[15:0]</td>
<td>BTP[15:0]</td>
<td>FT[15:0]</td>
<td>BT[15:0]</td>
<td>BTP[15:0]</td>
<td>FT[15:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>RTC_WPR</td>
<td>KEY[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x28</td>
<td>RTC_CALR</td>
<td>CALM[8:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2C</td>
<td>RTC_SHIFTR</td>
<td>ADD[14:0]</td>
<td>SUBFS[14:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x34</td>
<td>RTC_TSDR</td>
<td>WDU[1:0]</td>
<td>MT</td>
<td>MU[3:0]</td>
<td>DT [1:0]</td>
<td>DU[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x38</td>
<td>RTC_TSSSR</td>
<td>SS[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 144. RTC register map and reset values (continued)

| Offset | Register          | Reset value | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|-------------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x40   | RTC_ALRMAR        |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 |
| 0x44   | RTC_ALRMASSR      | MASKS[3:0]  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 |
| 0x48   | RTC_ALRMBR        | MASKS[3:0]  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 |
| 0x4C   | RTC_ALRMBSR       | MASKS[3:0]  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 |
| 0x50   | RTC_SR            |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 |
| 0x54   | RTC_MISR          |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 |
| 0x5C   | RTC_SCR           |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   | Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0  | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 | 0  | 0  | 0 |

Refer to Section 2.2 on page 54 for the register boundary addresses.
30 Tamper and backup registers (TAMP)

30.1 Introduction

5 32-bit backup registers are retained in all low-power modes and also in V_{BAT} mode. They can be used to store sensitive data as their content is protected by a tamper detection circuit. 2 tamper pins and 4 internal tampers are available for anti-tamper detection. The external tamper pins can be configured for edge detection, or level detection with or without filtering.

30.2 TAMP main features

- 5 backup registers:
  - the backup registers (TAMP_BKPxR) are implemented in the RTC domain that remains powered-on by V_{BAT} when the V_{DD} power is switched off.
- 2 external tamper detection events.
  - External passive tampers with configurable filter and internal pull-up.
- 4 internal tamper events.
- Any tamper detection can generate a RTC timestamp event.
- Any tamper detection can erase the backup registers.
30.3 TAMP functional description

30.3.1 TAMP block diagram

Figure 283. TAMP block diagram

1. The number of external and internal tampers depends on products.
30.3.2 TAMP pins and internal signals

Table 145. TAMP input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP_INx (x = pin index)</td>
<td>Input</td>
<td>Tamper input pin</td>
</tr>
</tbody>
</table>

Table 146. TAMP internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tamp_ker_ck</td>
<td>Input</td>
<td>TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLK in this document</td>
</tr>
<tr>
<td>tamp_pclk</td>
<td>Input</td>
<td>TAMP APB clock, connected to rtc_pclk</td>
</tr>
<tr>
<td>tamp_itamp[y]       (y = signal index)</td>
<td>Inputs</td>
<td>Internal tamper event sources</td>
</tr>
<tr>
<td>tamp_EVT</td>
<td>Output</td>
<td>Tamper event detection (internal or external) The tamp_EVT is used to generate a RTC timestamp event</td>
</tr>
<tr>
<td>tamp_erase</td>
<td>Output</td>
<td>Device secrets erase request following tamper event detection (internal or external)</td>
</tr>
<tr>
<td>tamp_IT</td>
<td>Output</td>
<td>TAMP interrupt (refer to Section 30.5: TAMP interrupts for details)</td>
</tr>
<tr>
<td>tamp_TX[y]           (y = signal index)</td>
<td>Output</td>
<td>Tamper detection trigger</td>
</tr>
</tbody>
</table>

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some detections modes are not available in some low-power modes or V_{BAT} when the selected clock is not LSE (refer to Section 30.4: TAMP low-power modes for more details).

Table 147. TAMP interconnection

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>tamp_EVT</td>
<td>rtc_tamp_EVT used to generate a timestamp event</td>
</tr>
<tr>
<td>tamp_erase</td>
<td>The tamp_erase signal is used to erase the device secrets listed hereafter: backup registers</td>
</tr>
<tr>
<td>tamp_ITamp3</td>
<td>LSE monitoring</td>
</tr>
<tr>
<td>tamp_ITamp4</td>
<td>HSE monitoring</td>
</tr>
<tr>
<td>tamp_ITamp5</td>
<td>RTC calendar overflow (rtc_calovf)</td>
</tr>
<tr>
<td>tamp_ITamp6</td>
<td>ST manufacturer readout</td>
</tr>
</tbody>
</table>

30.3.3 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable TAMP registers write access.
Tamper and backup registers (TAMP)

30.3.4 Tamper detection

The tamper detection can be configured for the following purposes:

- erase the backup registers (default configuration)
- generate an interrupt, capable to wakeup from Stop and Standby mode
- generate a hardware trigger for the low-power timers

TAMP backup registers

The backup registers (TAMP_BKPXR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers are reset when a tamper detection event occurs except if the TAMPxNOER bit is set, or if the TAMPxMSK is set in the TAMP_CR2 register.

Note: The backup registers are also erased when the readout protection of the flash is changed from level 1 to level 0.

Tamper detection initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

When TAMPxMSK is cleared:

- The TAMPxF flag is asserted after the tamper event on the pin, with the latency provided below:
  - 3 ck_apre cycles when TAMPFLT differs from 0x0 (level detection with filtering)
  - 3 ck_apre cycles when TAMPTS = 1 (timestamp on tamper event)
  - No latency when TAMPFLT = 0x0 (edge detection) and TAMPTS = 0

  A new tamper occurring on the same pin during this period and as long as TAMPxF is set cannot be detected.

When TAMPxMSK is set:

- A new tamper occurring on the same pin cannot be detected during the latency described above and 2.5 ck_rtc additional cycles.

  By setting the TAMPxE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit in cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration allows to trig automatically the low-power timers in Stop mode, without requiring the system wakeup to perform the TAMPxF clearing. In this case, the backup registers are not cleared.
This feature is available only when the tamper is configured in the Level detection with filtering on tamper inputs (passive mode) mode (TAMPFLT ≠ 00 and active mode is not selected).

**Timestamp on tamper event**

With TAMPTS set to 1 in the RTC_CR, any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register TAMPxF is set in the TAMP_SR at the same time that TSF or TSOVF is set in the RTC_SR.

**Edge detection on tamper inputs (passive mode)**

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge/high level or a falling edge/low level is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

**Caution:** When using the edge detection, it is recommended to check by software the tamper pin level just after enabling the tamper detection (by reading the GPIO registers), and before writing sensitive values in the backup registers, to ensure that an active edge did not occur before enabling the tamper event detection.

When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP_BKPXR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

**Note:** Tamper detection is still active when V_DD power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPx is mapped should be externally tied to the correct level.

**Level detection with filtering on tamper inputs (passive mode)**

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

**Note:** Refer to the datasheet for the electrical characteristics of the pull-up resistors.
30.4 TAMP low-power modes

Table 148. Effect of low-power modes on TAMP

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

30.5 TAMP interrupts

The interrupt channel is set in the interrupt status register. The interrupt output is also activated.

Table 149. Interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag(1)</th>
<th>Enable control bit(2)</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop and Standby modes</th>
<th>Exit from Shutdown mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP</td>
<td>Tamper x(3)</td>
<td>TAMPxF</td>
<td>TAMPxIE</td>
<td>Write 1 in CTAMPxF</td>
<td>Yes</td>
<td>Yes(4)</td>
<td>Yes(5)</td>
</tr>
<tr>
<td></td>
<td>Internal tamper y(3)</td>
<td>ITAMPyF</td>
<td>ITAMPyIE</td>
<td>Write 1 in CITAMPxF</td>
<td>Yes</td>
<td>Yes(4)</td>
<td>Yes(5)</td>
</tr>
</tbody>
</table>

1. The event flags are in the TAMP_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the TAMP_MISR register.
3. The number of tampers and internal tampers events depend on products.
4. In case of level detection with filtering passive tamper mode, wakeup from Stop and Standby modes is possible only when the TAMP clock source is LSE or LSI.
5. In case of level detection with filtering passive tamper mode, wakeup from Shutdown modes is possible only when the TAMP clock source is LSE.

30.6 TAMP registers

Refer to Section 1.2 on page 50 of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).
### 30.6.1 TAMP control register 1 (TAMP_CR1)

Address offset: 0x00  
RTC domain reset value: 0xFFFF 0000  
System reset: not affected

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

Bits 31:24 Reserved, must be kept at reset value.

- Bit 23 Reserved, must be kept at reset value.
- Bit 22 Reserved, must be kept at reset value.

- Bit 21 **ITAMP6E**: Internal tamper 6 enable: ST manufacturer readout  
  0: Internal tamper 6 disabled.  
  1: Internal tamper 6 enabled: a tamper is generated in case of ST manufacturer readout.

- Bit 20 **ITAMP5E**: Internal tamper 5 enable: RTC calendar overflow  
  0: Internal tamper 5 disabled.  
  1: Internal tamper 5 enabled: a tamper is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.

- Bit 19 **ITAMP4E**: Internal tamper 4 enable: HSE monitoring  
  0: Internal tamper 4 disabled.  
  1: Internal tamper 4 enabled: a tamper is generated when the HSE frequency is below or above thresholds.

- Bit 18 **ITAMP3E**: Internal tamper 3 enable: LSE monitoring  
  0: Internal tamper 3 disabled.  
  1: Internal tamper 3 enabled: a tamper is generated when the LSE frequency is below or above thresholds.

- Bit 17 Reserved, must be kept at reset value.
- Bit 16 Reserved, must be kept at reset value.

Bits 15:2 Reserved, must be kept at reset value.

- Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable  
  0: Tamper detection on TAMP_IN2 is disabled.  
  1: Tamper detection on TAMP_IN2 is enabled.

- Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable  
  0: Tamper detection on TAMP_IN1 is disabled.  
  1: Tamper detection on TAMP_IN1 is enabled.
30.6.2 Tamper control register 2 (TAMP_CR2)

Address offset: 0x04
RTC domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>TAMP2TRG</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>TAMP1TRG</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22:18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>TAMP2MSK</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TAMP1MSK</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
30.6.3 TAMP filter control register (TAMP_FLTCR)

Address offset: 0x0C
RTC domain reset value: 0x0000 0000
System reset: not affected

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

Bit 7 TAMPPUDIS: TAMP_INx pull-up disable
This bit determines if each of the TAMPx pins are precharged before each sample.
0: Precharge TAMP_INx pins before sampling (enable internal pull-up)
1: Disable precharge of TAMP_INx pins.
Notes:

This register concerns only the tamper inputs in passive mode.

30.6.4 TAMP interrupt enable register (TAMP_IER)

Address offset: 0x2C

RTC domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.
30.6.5 TAMP status register (TAMP_SR)

Address offset: 0x30

RTC domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 ITAMP6F: ST manufacturer readout tamper detection flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.
Bit 20 **ITAMP5F**: RTC calendar overflow tamper detection flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 **ITAMP4F**: HSE monitoring tamper detection flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.

Bit 18 **ITAMP3F**: LSE monitoring tamper detection flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **TAMP2F**: TAMP2 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.

Bit 0 **TAMP1F**: TAMP1 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.

### 30.6.6 TAMP masked interrupt status register (TAMP_MISR)

Address offset: 0x34
RTC domain reset value: 0x0000 0000
System reset: not affected

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

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6MF**: ST manufacturer readout tamper interrupt masked flag
This flag is set by hardware when the internal tamper 6 interrupt is raised.

Bit 20 **ITAMP5MF**: RTC calendar overflow tamper interrupt masked flag
This flag is set by hardware when the internal tamper 5 interrupt is raised.

Bit 19 **ITAMP4MF**: HSE monitoring tamper interrupt masked flag
This flag is set by hardware when the internal tamper 4 interrupt is raised.

Bit 18 **ITAMP3MF**: LSE monitoring tamper interrupt masked flag
This flag is set by hardware when the internal tamper 3 interrupt is raised.

Bit 17 Reserved, must be kept at reset value.
Tamper and backup registers (TAMP) RM0444

30.6.7 TAMP status clear register (TAMP_SCR)

Address offset: 0x3C
System reset value: 0x0000 0000

Bits 31:24 Reserved, must be kept at reset value.
Bit 23 Reserved, must be kept at reset value.
Bit 22 Reserved, must be kept at reset value.
Bit 21 CITAMP6F: Clear ITAMP6 detection flag
Writing 1 in this bit clears the ITAMP6F bit in the TAMP_SR register.
Bit 20 CITAMP5F: Clear ITAMP5 detection flag
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.
Bit 19 CITAMP4F: Clear ITAMP4 detection flag
Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.
Bit 18 CITAMP3F: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.
Bit 17 Reserved, must be kept at reset value.
Bit 16 Reserved, must be kept at reset value.
Bits 15:2 Reserved, must be kept at reset value.
Bit 1 CITAMP2F: Clear TAMP2 detection flag
Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.
Bit 0 CITAMP1F: Clear TAMP1 detection flag
Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.
30.6.8 TAMPER backup x register (TAMP_BKPxR)

Address offset: 0x100 + 0x04 * x, (x = 0 to 4)
Backup domain reset value: 0x0000 0000
System reset: not affected

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

Bits 31:0 BKP[31:0]

The application can write or read data to and from these registers. They are powered-on by VBAT when VDD is switched off, so that they are not reset by System reset, and their contents remain valid when the device operates in low-power mode.

In the default configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.
30.6.9 TAMP register map

| Offset | Register     | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | TAMP_CR1     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  | 1  | 1  | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | TAMP_CR2     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | TAMP_FLTCR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2C   | TAMP_IER     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x30   | TAMP_SR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x34   | TAMP_MISR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x3C   | TAMP_SCR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x100 + 0x04*x, (x = 0 to 4) | TAMP_BKPxR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.2 on page 54 for the register boundary addresses.
31 Inter-integrated circuit (I2C) interface

31.1 Introduction

The I²C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I²C bus. It provides multimaster capability, and controls all I²C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

It is also SMBus (system management bus) and PMBus (power management bus) compatible.

DMA can be used to reduce CPU overload.

31.2 I2C main features

- I²C bus specification rev03 compatibility:
  - Slave and master modes
  - Multimaster capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
The following additional features are also available depending on the product implementation (see Section 31.3: I2C implementation):

- SMBus specification rev 3.0 compatibility:
  - Hardware PEC (Packet Error Checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and Device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock: a choice of independent clock sources allowing the I2C communication speed to be independent from the PCLK reprogramming
- Wakeup from Stop mode on address match.

### 31.3 I2C implementation

This manual describes the full set of features implemented in I2C1 and I2C2. I2C2 supports a smaller set of features, but is otherwise identical to I2C1. The differences are listed below.

<table>
<thead>
<tr>
<th>I2C features(1)</th>
<th>I2C1</th>
<th>I2C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-bit addressing mode</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>10-bit addressing mode</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Standard mode (up to 100 kbit/s)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Fast mode (up to 400 kbit/s)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Fast-mode Plus with 20mA output drive I/Os (up to 1 Mbit/s)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Independent clock</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Wakeup from Stop mode</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>SMBus/PMbus</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

1. X = supported.

### 31.4 I2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I2C bus.

This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).
If SMBus feature is supported: the additional optional SMBus Alert pin (SMBA) is also available.

31.4.1 I2C1 block diagram

The block diagram of the I2C1 interface is shown in Figure 284.

Figure 284. I2C1 block diagram

The I2C1 is clocked by an independent clock source which allows to the I2C to operate independently from the PCLK frequency.

For I2C I/Os supporting 20 mA output current drive for Fast-mode Plus operation, the driving capability is enabled through control bits in the system configuration controller (SYSCFG). Refer to Section 31.3: I2C implementation.
### 31.4.2 I2C2 block diagram

The block diagram of the I2C interface is shown in Figure 285.

**Figure 285. I2C2 block diagram**

For I2C I/Os supporting 20 mA output current drive for Fast-mode Plus operation, the driving capability is enabled through control bits in the system configuration controller (SYSCFG). Refer to Section 31.3: I2C implementation.

### 31.4.3 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period $t_{I2CCLK}$ must respect the following conditions:

$$t_{I2CCLK} < \left( (t_{LOW} - t_{filters}) / 4 \right) \quad \text{and} \quad t_{I2CCLK} < t_{HIGH}$$
with:

\( t_{LOW} \): SCL low time and \( t_{HIGH} \): SCL high time

\( t_{filters} \): when enabled, sum of the delays brought by the analog filter and by the digital filter.

Analog filter delay is maximum 260 ns. Digital filter delay is \( DNF \times t_{I2CCLK} \).

The PCLK clock period \( t_{PCLK} \) must respect the following condition:

\[ t_{PCLK} < \frac{4}{3} t_{SCL} \]

with \( t_{SCL} \): SCL period

**Caution:** When the I2C kernel is clocked by PCLK, this clock must respect the conditions for \( t_{I2CCLK} \).

### 31.4.4 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

**Communication flow**

In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.
Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

### 31.4.5 I2C initialization

#### Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller. Then the I2C can be enabled by setting the PE bit in the I2C_CR1 register.

When the I2C is disabled (PE=0), the I2C performs a software reset. Refer to Section 31.4.6: Software reset for more details.

#### Noise filters

Before enabling the I2C peripheral by setting the PE bit in I2C_CR1 register, the user must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I2C specification which requires the suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. The user can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x I2C peripheral clock periods. This allows to suppress spikes with a programmable length of 1 to 15 I2CCLK periods.

#### Table 152. Comparison of analog vs. digital filters

<table>
<thead>
<tr>
<th></th>
<th>Analog filter</th>
<th>Digital filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse width of suppressed spikes</td>
<td>≥ 50 ns</td>
<td>Programmable length from 1 to 15 I2C peripheral clocks</td>
</tr>
<tr>
<td>Benefits</td>
<td>Available in Stop mode</td>
<td>– Programmable length: extra filtering capability vs. standard requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Stable length</td>
</tr>
<tr>
<td>Drawbacks</td>
<td>Variation vs. temperature, voltage, process</td>
<td>Wakeup from Stop mode on address match is not available when digital filter is enabled</td>
</tr>
</tbody>
</table>
Caution: Changing the filter configuration is not allowed when the I2C is enabled.

**I2C timings**

The timings must be configured in order to guarantee a correct data hold and setup time, used in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C configuration window.

![Figure 287. Setup and hold timings](image-url)
• When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is \( t_{SDADEL} = SDADEL \times t_{PRESC} + t_{I2CCLK} \) where \( t_{PRESC} = (PRESC+1) \times t_{I2CCLK} \).

\( t_{SDADEL} \) impacts the hold time \( t_{HD;DAT} \).

The total SDA output delay is:

\[ t_{SYNC1} + \{ (SDADEL \times (PRESC+1) + 1) \times t_{I2CCLK} \} \]

\( t_{SYNC1} \) duration depends on these parameters:

- SCL falling slope
- When enabled, input delay brought by the analog filter: \( t_{AF(min)} < t_{AF} < t_{AF(max)} \)
- When enabled, input delay brought by the digital filter: \( t_{DNF} = DNF \times t_{I2CCLK} \)
- Delay due to SCL synchronization to I2CCLK clock (2 to 3 I2CCLK periods)

In order to bridge the undefined region of the SCL falling edge, the user must program SDADEL in such a way that:

\( \{ t_{I_{(max)}} + t_{HD;DAT_{(min)}} - t_{AF_{(min)}} - \left[ (DNF+3) \times t_{I2CCLK} \right] \} / \left( (PRESC+1) \times t_{I2CCLK} \right) \leq SDADEL \)

\( SDADEL \leq \{ t_{VD;DAT_{(max)}} - t_{AF_{(max)}} - \left[ (DNF+4) \times t_{I2CCLK} \right] \} / \left( (PRESC+1) \times t_{I2CCLK} \right) \)

Note: \( t_{AF_{(min)}} / t_{AF_{(max)}} \) are part of the equation only when the analog filter is enabled. Refer to device datasheet for \( t_{AF} \) values.

The maximum \( t_{HD;DAT} \) can be 3.45 µs, 0.9 µs and 0.45 µs for Standard-mode, Fast-mode and Fast-mode Plus, but must be less than the maximum of \( t_{VD;DAT} \) by a transition time. This maximum must only be met if the device does not stretch the LOW period \( t_{LOW} \) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case, so in this case the previous equation becomes:

\( SDADEL \leq \{ t_{VD;DAT_{(max)}} - t_{I_{(max)}} - 260 \text{ ns} - \left[ (DNF+4) \times t_{I2CCLK} \right] \} / \left( (PRESC+1) \times t_{I2CCLK} \right) \)

Note: This condition can be violated when NOSTRETCH=0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL value.

Refer to Table 153: I2C-SMBUS specification data setup and hold times for \( t_{I} \), \( t_{r} \), \( t_{HD;DAT} \) and \( t_{VD;DAT} \) standard values.

- After \( t_{SDADEL} \) delay, or after sending SDA output in case the slave had to stretch the clock because the data was not yet written in I2C_TXDR register, SCL line is kept at low level during the setup time. This setup time is \( t_{SCLDEL} = (SCLDEL+1) \times t_{I2CCLK} \) where \( t_{I2CCLK} = (PRESC+1) \times t_{I2CCLK} \).

\( t_{SCLDEL} \) impacts the setup time \( t_{SU;DAT} \).

In order to bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL in such a way that:

\( \{ t_{I_{(max)}} + t_{SU;DAT_{(min)}} \} / \left( (PRESC+1) \times t_{I2CCLK} \right) - 1 \leq SCLDEL \)

Refer to Table 153: I2C-SMBUS specification data setup and hold times for \( t_{I} \) and \( t_{SU;DAT} \) standard values.

The SDA and SCL transition time values to be used are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature whatever the application.
At every clock pulse, after SCL falling edge detection, the I2C master or slave stretches SCL low during at least \([(SDADEL+SCLDEL+1) \times (PRESC+1) + 1] \times t_{I2CCLK}\) in both transmission and reception modes. In transmission mode, in case the data is not yet written in I2C_TXDR when SDADEL counter is finished, the I2C keeps on stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

If NOSTRETCH=1 in slave mode, the SCL is not stretched. Consequently the SDADEL must be programmed in such a way to guarantee also a sufficient setup time.

Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is \(t_{SCLL} = (SCLL+1) \times t_{PRESC}\) where \(t_{PRESC} = (PRESC+1) \times t_{I2CCLK}\). \(t_{SCLL}\) impacts the SCL low time \(t_{LOW}\).
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is \(t_{SCLH} = (SCLH+1) \times t_{PRESC}\) where \(t_{PRESC} = (PRESC+1) \times t_{I2CCLK}\). \(t_{SCLH}\) impacts the SCL high time \(t_{HIGH}\).

Refer to I2C master initialization for more details.

Caution: Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral. Refer to I2C slave initialization for more details.

Caution: Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.
31.4.6 Software reset

A software reset can be performed by clearing the PE bit in the I2C_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:

1. I2C_CR2 register: START, STOP, NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:

1. I2C_CR2 register: PECBYTE
2. I2C_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least 3 APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence: - Write PE=0 - Check PE=0 - Write PE=1.
31.4.7 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the 8th SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE=0). If RXNE=1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the 8th and 9th SCL pulse (before the Acknowledge pulse).
Transmission

If the I2C_TXDR register is not empty (TXE=0), its content is copied into the shift register after the 9th SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE=1, meaning that no data is written yet in I2C_TXDR, SCL line is stretched low until I2C_TXDR is written. The stretch is done after the 9th SCL pulse.

![Figure 290. Data transmission](image)

Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in master mode. By default it is disabled in slave mode, but it can be enabled by software by setting the SBC (Slave Byte Control) bit in the I2C_CR2 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this mode, TCR flag is set when the number of bytes programmed in NBYTES has been transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.
When RELOAD=0 in master mode, the counter can be used in 2 modes:

- **Automatic end mode** (AUTOEND = ‘1’ in the I2C_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred.

- **Software end mode** (AUTOEND = ‘0’ in the I2C_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2C_CR2 register. This mode must be used when the master wants to send a RESTART condition.

**Caution:** The AUTOEND bit has no effect when the RELOAD bit is set.

### Table 154. I2C configuration

<table>
<thead>
<tr>
<th>Function</th>
<th>SBC bit</th>
<th>RELOAD bit</th>
<th>AUTOEND bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Tx/Rx NBYTES + STOP</td>
<td>x</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Master Tx/Rx + NBYTES + RESTART</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Slave Tx/Rx</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>all received bytes ACKed</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Slave Rx with ACK control</td>
<td>1</td>
<td>1</td>
<td>x</td>
</tr>
</tbody>
</table>

### 31.4.8 I2C slave mode

#### I2C slave initialization

In order to work in slave mode, the user must enable at least one slave address. Two registers I2C_OAR1 and I2C_OAR2 are available in order to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C_OAR1 register.
  
  OA1 is enabled by setting the OA1EN bit in the I2C_OAR1 register.

- If additional slave addresses are required, the 2nd slave address OA2 can be configured. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK=7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

  These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK=0.

  OA2 is enabled by setting the OA2EN bit in the I2C_OAR2 register.

- The General Call address is enabled by setting the GCEN bit in the I2C_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.
By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, in order to perform software actions. If the master does not support clock stretching, the I2C must be configured with NOSTRETCH=1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled the user must read the ADDCODE[6:0] bits in the I2C_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

**Slave clock stretching (NOSTRETCH = 0)**

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C_TXDR register, or if the first data byte is not written when the ADDR flag is cleared (TXE=1). This stretch is released when the data is written to the I2C_TXDR register.
- In reception when the I2C_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C_RXDR is read.
- When TCR = 1 in Slave Byte Control mode, reload mode (SBC=1 and RELOAD=1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTE[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during \([(SDADEL+SCLDEL+1) \times (PRESC+1) + 1] \times \frac{1}{I2CLK}\)

**Slave without clock stretching (NOSTRETCH = 1)**

When NOSTRETCH = 1 in the I2C_CR1 register, the I2C slave does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, he ensures that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2C_RXDR register before the 9th SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.
Slave Byte Control mode

In order to allow byte ACK control in slave reception mode, Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. This is required to be compliant with SMBus standards.

Reload mode must be selected in order to allow byte ACK control in slave reception mode (RELOAD=1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the 8th and 9th SCL pulses. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

Note: The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR=1.

The RELOAD bit value can be changed when ADDR=1, or when TCR=1.

Caution: Slave Byte Control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH=1 is not allowed.

---

**Figure 291. Slave initialization flowchart**

![Slave initialization flowchart](image-url)
Slave transmitter

A transmit interrupt status (TXIS) is generated when the I2C.TxDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C_Cr1 register.

The TXIS bit is cleared when the I2C.TxDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C_Cr1 register. The slave automatically releases the SCL and SDA lines in order to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C_Cr1 register, the STOPF flag is set in the I2C_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to ‘0’. In this case, if TXE = 0 when the slave address is received (ADDR=1), the user can choose either to send the content of the I2C.TxDR register as the first data byte, or to flush the I2C.TxDR register by setting the TXE bit in order to program a new data byte.

In Slave Byte Control mode (SBC=1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR=1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

Caution: When NOSTRETCH=1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C.TxDR register content in the ADDR subroutine, in order to program the first data byte. The first data byte to be sent must be previously programmed in the I2C.TxDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C.TxDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

If a TXIS event is needed, (Transmit Interrupt or Transmit DMA request), the user must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.
Figure 292. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=0

Slave transmission

Slave initialization

No

i2c_ISR.ADDR =1?

Yes

Read ADDCODE and DIR in i2C_ISR
Optional: Set i2C_ISR.TXE = 1
Set i2C_ICR.ADDRCF

No

SCL stretched

Yes

i2C_ISR.TXIS =1?

No

Write i2C_TXDR.TXDATA

Yes
Figure 293. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=1

Slave initialization

Slave transmission

I2C_ISR.TXIS = 1?

Yes

Write I2C_TXDR.TXDATA

No

I2C_ISR.STOPF = 1?

Yes

Optional: Set I2C_ISR.TXE = 1 and I2C_ISR.TXIS=1

Set I2C_ICR.STOPCF

No
Figure 294. Transfer bus diagrams for I2C slave transmitter

Example I2C slave transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes without 1st data flush, NOSTRETCH=0:

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes, NOSTRETCH=1:

EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

legend:
- transmission
- reception
- SCL stretch
Slave receiver

RXNE is set in I2C_ISR when the I2C_RXDR is full, and generates an interrupt if RXIE is set in I2C_CR1. RXNE is cleared when I2C_RXDR is read.

When a STOP is received and STOPIE is set in I2C_CR1, STOPF is set in I2C_ISR and an interrupt is generated.

Figure 295. Transfer sequence flowchart for slave receiver with NOSTRETCH=0

Slave reception

Slave initialization

I2C_ISR.ADDR = 1?

Read ADDCODE and DIR in I2C_ISR
Set I2C_ICR.ADDRCF

I2C_ISR.RXNE = 1?

Write I2C_RXDR.RXDATA

SCL stretched
Figure 296. Transfer sequence flowchart for slave receiver with NOSTRETCH=1

Slave reception

Slave initialization

I2C_ISR.RXNE =1?

No

Yes

Read I2C_RXDR.RXDATA

I2C_ISR.STOPF =1?

No

Yes

Set I2C_ICR.STOPCF

Figure 297. Transfer bus diagrams for I2C slave receiver

Example I2C slave receiver 3 bytes, NOSTRETCH=0:

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: RXNE ISR: rd data1
EV3: RXNE ISR: rd data2
EV4: RXNE ISR: rd data3

Example I2C slave receiver 3 bytes, NOSTRETCH=1:

EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2
EV3: RXNE ISR: rd data3
EV4: STOPF ISR: set STOPCF
31.4.9 I2C master mode

I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

A clock synchronization mechanism is implemented in order to support multi-master environment and slave clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a $t_{SYNC1}$ delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bits in the I2C_TIMINGR register.

The I2C detects its own SCL high level after a $t_{SYNC2}$ delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLH counter is reached reaches the value programmed in the SCLH[7:0] bits in the I2C_TIMINGR register.

Consequently the master clock period is:

$$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \left[\left( (SCLH+1) + (SCLL+1) \right) \times \left( (PRES+1) \times t_{I2CCLK} \right) \right]$$

The duration of $t_{SYNC1}$ depends on these parameters:

- SCL falling slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF $\times t_{I2CCLK}$
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

The duration of $t_{SYNC2}$ depends on these parameters:

- SCL rising slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF $\times t_{I2CCLK}$
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)
Caution: In order to be \(^2\)C or SMBus compliant, the master clock must respect the timings given below:
Inter-integrated circuit (I2C) interface

Table 155. I²C-SMBUS specification clock timings

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
<th>SMBUS</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>tSCL</td>
<td>SCL clock frequency</td>
<td>-</td>
<td>100</td>
<td>400</td>
<td>1000</td>
<td>100  kHz</td>
</tr>
<tr>
<td>tHD:STA</td>
<td>Hold time (repeated) START condition</td>
<td>4.0</td>
<td>-</td>
<td>0.6</td>
<td>0.26</td>
<td>4.0  µs</td>
</tr>
<tr>
<td>tSU:STA</td>
<td>Set-up time for a repeated START condition</td>
<td>4.7</td>
<td>-</td>
<td>0.6</td>
<td>0.26</td>
<td>4.7  µs</td>
</tr>
<tr>
<td>tSU:STO</td>
<td>Set-up time for STOP condition</td>
<td>4.0</td>
<td>-</td>
<td>0.6</td>
<td>0.26</td>
<td>4.0  µs</td>
</tr>
<tr>
<td>tBUF</td>
<td>Bus free time between a STOP and START condition</td>
<td>4.7</td>
<td>-</td>
<td>1.3</td>
<td>0.5</td>
<td>4.7  µs</td>
</tr>
<tr>
<td>tLOW</td>
<td>Low period of the SCL clock</td>
<td>4.7</td>
<td>-</td>
<td>1.3</td>
<td>0.5</td>
<td>4.7  µs</td>
</tr>
<tr>
<td>tHIGH</td>
<td>Period of the SCL clock</td>
<td>4.0</td>
<td>-</td>
<td>0.6</td>
<td>0.26</td>
<td>4.0  50 µs</td>
</tr>
<tr>
<td>tᵣ</td>
<td>Rise time of both SDA and SCL signals</td>
<td>-</td>
<td>1000</td>
<td>-</td>
<td>300</td>
<td>-     1200 ns</td>
</tr>
<tr>
<td>tᵫ</td>
<td>Fall time of both SDA and SCL signals</td>
<td>-</td>
<td>300</td>
<td>-</td>
<td>120</td>
<td>-     300 ns</td>
</tr>
</tbody>
</table>

Note:  
- SCLL is also used to generate the $t_{BUF}$ and $t_{SU:STA}$ timings.  
- SCLH is also used to generate the $t_{HD:STA}$ and $t_{SU:STO}$ timings.  
- Refer to Section 31.4.10: I2C_TIMINGR register configuration examples for examples of I2C_TIMINGR settings vs. I2CCLK frequency.

Master communication initialization (address phase)

In order to initiate the communication, the user must program the following parameters for the addressed slave in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of $t_{BUF}$.

In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

Note:  
The START bit is reset by hardware when the slave address has been sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs. In 10-bit addressing mode, when the Slave Address first 7 bits is NACKed by the slave, the...
Master re-launches automatically the slave address transmission until ACK is received. In this case ADDRCF must be set if a NACK is received from the slave, in order to stop sending the slave address.

If the I2C is addressed as a slave (ADDR=1) while the START bit is set, the I2C switches to slave mode and the START bit is cleared, when the ADDRCF bit is set.

**Note:** The same procedure is applied for a Repeated Start condition. In this case BUSY=1.

**Figure 299. Master initialization flowchart**

**Initialization of a master receiver addressing a 10-bit address slave**

- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set:
  (Re)Start + Slave address 10-bit header Write + Slave address 2nd byte + REStart + Slave address 10-bit header Read

**Figure 300. 10-bit address read access with HEAD10R=0**
If the master addresses a 10-bit address slave, transmits data to this slave and then reads data from the same slave, a master transmission flow must be done first. Then a repeated start is set with the 10 bit slave address configured with HEAD10R=1. In this case the master sends this sequence: ReStart + Slave address 10-bit header Read.

**Figure 301. 10-bit address read access with HEAD10R=1**

### Master transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the 9th SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C_CR1 register. The flag is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD=0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND=1), a STOP is automatically sent.
  - In software end mode (AUTOEND=0), the TC flag is set and the SCL line is stretched low in order to perform software actions:
    - A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.
    - A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. The NACKF flag is set in the I2C_ISR register, and an interrupt is generated if the NACKIE bit is set.
Figure 302. Transfer sequence flowchart for I2C master transmitter for $N \leq 255$ bytes

Master transmission

Master initialization

$N\text{BYTES} = N$
AUTOEND = 0 for RESTART, 1 for STOP
Configure slave address
Set I2C_CR2.START

...
Figure 303. Transfer sequence flowchart for I2C master transmitter for N>255 bytes

Master initialization

NBYTES = 0xFF; N=N-255
RELOAD = 1
Configure slave address
Set I2C_CR2.START

I2C_ISR.NACKF = 1?
Yes
End
No

I2C_ISR.TXIS = 1?
Yes
Write I2C_TXDR
No

I2C_ISR.TC = 1?
Yes
No

IF N< 256
NBYTES = N; N = 0; RELOAD = 0
AUTOEND = 0 for RESTART; 1 for STOP
ELSE
NBYTES = 0xFF; N = N-255
RELOAD = 1

NBYTES transmitted ?
Yes
Set I2C_CR2.START with slave address NBYTES ...
No

Set I2C_CR2.START with slave address NBYTES ...

End
Figure 304. Transfer bus diagrams for I2C master transmitter

Example I2C master transmitter 2 bytes, automatic end mode (STOP)

INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2

Example I2C master transmitter 2 bytes, software end mode (RESTART)

INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START
Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the 8th SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C_CR1 register. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD=0 and NBYTES[7:0] data have been transferred:
  - In automatic end mode (AUTOEND=1), a NACK and a STOP are automatically sent after the last received byte.
  - In software end mode (AUTOEND=0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:
    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.
    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
Figure 305. Transfer sequence flowchart for I2C master receiver for \( N \leq 255 \) bytes

1. **Master reception**
2. **Master initialization**
3. **\( N\)BYTES = \( N \)**
   AUTOEND = 0 for RESTART; 1 for STOP
   Configure slave address
   Set I2C_CR2.START
4. **\( I2C\_ISR.RXNE = 1? \)**
   - **No**
   - **Yes**
5. **Read I2C_RXDR**
6. **\( N\)BYTES received?**
   - **No**
   - **Yes**
7. **\( I2C\_ISR.TC = 1? \)**
   - **No**
   - **Yes**
   - **Set I2C_CR2.START with slave address \( N\)BYTES ...**
8. **End**
Figure 306. Transfer sequence flowchart for I2C master receiver for N >255 bytes

Master initialization

NBYTES = 0xFF; N=N-255
RELOAD =1
Configure slave address
Set I2C_CR2.START

I2C_ISR.RXNE =1?
No
Yes
Read I2C_RXDR

NBYTES received?
No
Yes
NBYTES =0xFF; N=N-255
RELOAD =1

I2C_ISR.TC = 1?
No
Yes
Set I2C_CR2.START with slave address NBYTES ...

I2C_ISR.TCR = 1?
No
Yes
IF N< 256
NBYTES =N; N=0;RELOAD=0
AUTOEND=0 for RESTART; 1 for STOP
ELSE
NBYTES =0xFF;N=N-255
RELOAD=1

End
Example I2C master receiver 2 bytes, automatic end mode (STOP)

INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2

Example I2C master receiver 2 bytes, software end mode (RESTART)

INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: read data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START
### 31.4.10 I2C_TIMINGR register configuration examples

The tables below provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I2C specification. In order to get more accurate configuration values, the STM32CubeMX tool (I2C Configuration window) must be used.

#### Table 156. Examples of timing settings for \( f_{I2CCLK} = 8 \text{ MHz} \)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 kHz</td>
<td>100 kHz</td>
<td>400 kHz</td>
</tr>
<tr>
<td>PRESC</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SCLL</td>
<td>0xC7</td>
<td>0x13</td>
<td>0x9</td>
</tr>
<tr>
<td>( t_{SCLL} )</td>
<td>200\times250 ns = 50 µs</td>
<td>20\times250 ns = 5.0 µs</td>
<td>10\times125 ns = 1250 ns</td>
</tr>
<tr>
<td>SCLH</td>
<td>0xC3</td>
<td>0xF</td>
<td>0x3</td>
</tr>
<tr>
<td>( t_{SCLH} )</td>
<td>196\times250 ns = 49 µs</td>
<td>16\times250 ns = 4.0µs</td>
<td>4\times125 ns = 500ns</td>
</tr>
<tr>
<td>( t_{SCL} )</td>
<td>~100 µs(^{(1)})</td>
<td>~10 µs(^{(2)})</td>
<td>~2500 ns(^{(3)})</td>
</tr>
<tr>
<td>SDADEL</td>
<td>0x2</td>
<td>0x2</td>
<td>0x1</td>
</tr>
<tr>
<td>( t_{SDADEL} )</td>
<td>2\times250 ns = 500 ns</td>
<td>2\times250 ns = 500 ns</td>
<td>1\times125 ns = 125 ns</td>
</tr>
<tr>
<td>SCLDEL</td>
<td>0x4</td>
<td>0x4</td>
<td>0x3</td>
</tr>
<tr>
<td>( t_{SCLDEL} )</td>
<td>5\times250 ns = 1250 ns</td>
<td>5\times250 ns = 1250 ns</td>
<td>4\times125 ns = 500 ns</td>
</tr>
</tbody>
</table>

1. \( t_{SCL} \) period is greater than \( t_{SCLL} + t_{SCLH} \) due to SCL internal detection delay. Values provided for \( t_{SCL} \) are examples only.
2. \( t_{SYNC1} + t_{SYNC2} \) minimum value is \( 4 \times t_{I2CCCLK} = 500 \) ns. Example with \( t_{SYNC1} + t_{SYNC2} = 1000 \) ns.
3. \( t_{SYNC1} + t_{SYNC2} \) minimum value is \( 4 \times t_{I2CCCLK} = 500 \) ns. Example with \( t_{SYNC1} + t_{SYNC2} = 750 \) ns.
4. \( t_{SYNC1} + t_{SYNC2} \) minimum value is \( 4 \times t_{I2CCCLK} = 500 \) ns. Example with \( t_{SYNC1} + t_{SYNC2} = 655 \) ns.

#### Table 157. Examples of timing settings for \( f_{I2CCLK} = 16 \text{ MHz} \)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 kHz</td>
<td>100 kHz</td>
<td>400 kHz</td>
</tr>
<tr>
<td>PRESC</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SCLL</td>
<td>0xC7</td>
<td>0x13</td>
<td>0x9</td>
</tr>
<tr>
<td>( t_{SCLL} )</td>
<td>200 \times 250 ns = 50 µs</td>
<td>20 \times 250 ns = 5.0 µs</td>
<td>10 \times 125 ns = 1250 ns</td>
</tr>
<tr>
<td>SCLH</td>
<td>0xC3</td>
<td>0xF</td>
<td>0x3</td>
</tr>
<tr>
<td>( t_{SCLH} )</td>
<td>196 \times 250 ns = 49 µs</td>
<td>16 \times 250 ns = 4.0µs</td>
<td>4 \times 125 ns = 500ns</td>
</tr>
<tr>
<td>( t_{SCL} )</td>
<td>~100 µs(^{(1)})</td>
<td>~10 µs(^{(2)})</td>
<td>~2500 ns(^{(3)})</td>
</tr>
<tr>
<td>SDADEL</td>
<td>0x2</td>
<td>0x2</td>
<td>0x2</td>
</tr>
<tr>
<td>( t_{SDADEL} )</td>
<td>2 \times 250 ns = 500 ns</td>
<td>2 \times 250 ns = 500 ns</td>
<td>2 \times 125 ns = 250 ns</td>
</tr>
<tr>
<td>SCLDEL</td>
<td>0x4</td>
<td>0x4</td>
<td>0x3</td>
</tr>
<tr>
<td>( t_{SCLDEL} )</td>
<td>5 \times 250 ns = 1250 ns</td>
<td>5 \times 250 ns = 1250 ns</td>
<td>4 \times 125 ns = 500 ns</td>
</tr>
</tbody>
</table>

1. \( t_{SCL} \) period is greater than \( t_{SCLL} + t_{SCLH} \) due to SCL internal detection delay. Values provided for \( t_{SCL} \) are examples only.
31.4.11 SMBus specific features

This section is relevant only when SMBus feature is supported. Refer to Section 31.3: I2C implementation.

Introduction

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on \^I^2^C\ principles of operation. SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBUS specification (http://smbus.org).

The System Management Bus Specification refers to three types of devices.

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as master or slave device, and also as a host.
Bus protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write and Block Write-Block Read Process Call. These protocols should be implemented by the user software.

For more details of these protocols, refer to SMBus specification (http://smbus.org).

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C_CR1 register. The ARP commands should be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus Address Resolution Protocol, refer to SMBus specification (http://smbus.org).

Received Command and Data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2C_CR1 register. Refer to Slave Byte Control mode on page 897 for more details.

Host Notify protocol

This peripheral supports the Host Notify protocol by setting the SMBHEN bit in the I2C_CR1 register. In this case the host acknowledges the SMBus Host address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

SMBus alert

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the Alert Response Address (0b0001 100). Only the device(s) which pulled SMBALERT# low acknowledges the Alert Response Address.

When configured as a slave device(SMBHEN=0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN=1), the ALERT flag is set in the I2C_ISR register when a falling edge is detected on the SMBA pin and ALERTEN=1. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. When ALERTEN=0, the ALERT line is considered high even if the external SMBA pin is low.

If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN=0.
Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. Packet Error Checking is implemented by appending a Packet Error Code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows to send a Not Acknowledge automatically when the received byte does not match with the hardware calculated PEC.

Timeouts

This peripheral embeds hardware timers in order to be compliant with the 3 timeouts defined in SMBus specification.

Table 159. SMBus timeout specifications

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{\text{T\text{IMEOUT}}}$</td>
<td>Detect clock low timeout</td>
<td>25</td>
<td>35 ms</td>
</tr>
<tr>
<td>$t_{\text{LOW:\text{SEXT}}}^{(1)}$</td>
<td>Cumulative clock low extend time (slave device)</td>
<td>-</td>
<td>25 ms</td>
</tr>
<tr>
<td>$t_{\text{LOW:\text{MEXT}}}^{(2)}$</td>
<td>Cumulative clock low extend time (master device)</td>
<td>-</td>
<td>10 ms</td>
</tr>
</tbody>
</table>

1. $t_{\text{LOW:\text{SEXT}}}$ is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that, another slave device or the master also extends the clock causing the combined clock low extend time to be greater than $t_{\text{LOW:\text{SEXT}}}$. Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.

2. $t_{\text{LOW:\text{MEXT}}}$ is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master also extends the clock causing the combined clock low time to be greater than $t_{\text{LOW:\text{MEXT}}}$ on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.

Figure 308. Timeout intervals for $t_{\text{LOW:\text{SEXT}}}$, $t_{\text{LOW:\text{MEXT}}}$
Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for $t_{\text{IDLE}}$ greater than $t_{\text{HIGH,MAX}}$. (refer to Table 153: I2C-SMBUS specification data setup and hold times)

This timing parameter covers the condition where a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

31.4.12 SMBus initialization

This section is relevant only when SMBus feature is supported. Refer to Section 31.3: I2C implementation.

In addition to I2C initialization, some other specific initialization must be done in order to perform SMBus communication:

Received Command and Data Acknowledge control (Slave mode)

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in slave mode, the Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. Refer to Slave Byte Control mode on page 897 for more details.

Specific address (Slave mode)

The specific SMBus addresses must be enabled if needed. Refer to Bus idle detection on page 920 for more details.

- The SMBus Device Default address (0b1100 001) is enabled by setting the SMBDEN bit in the I2C_CR1 register.
- The SMBus Host address (0b0001 000) is enabled by setting the SMBHEN bit in the I2C_CR1 register.
- The Alert Response Address (0b0001100) is enabled by setting the ALERTEN bit in the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit in the I2C_CR1 register. Then the PEC transfer is managed with the help of a hardware byte counter: NBYTES[7:0] in the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in slave mode. The PEC is transferred after NBYTES-1 data have been transferred when the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C is enabled.
Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits in the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

- **tTIMEOUT check**
  
  In order to enable the tTIMEOUT check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value in order to check the tTIMEOUT parameter. The TIDLE bit must be configured to '0' in order to detect the SCL low level timeout.
  
  Then the timer is enabled by setting the TIMOUTEN in the I2C_TIMEOUTR register. If SCL is tied low for a time greater than (TIMEOUTA+1) x 2048 x tI2CCLK, the TIMEOUT flag is set in the I2C_ISR register.
  
  Refer to Table 161: Examples of TIMEOUTA settings for various I2CCLK frequencies (max tTIMEOUT = 25 ms).

**Caution:** Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMOUTEN bit is set.

- **tLOW:SEXT and tLOW:MEXT check**
  
  Depending on if the peripheral is configured as a master or as a slave, The 12-bit TIMEOUTB timer must be configured in order to check tLOW:SEXT for a slave and tLOW:MEXT for a master. As the standard specifies only a maximum, the user can choose the same value for both.
  
  Then the timer is enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register. If the SMBus peripheral performs a cumulative SCL stretch for a time greater than (TIMEOUTB+1) x 2048 x tI2CCLK, and in the timeout interval described in Bus idle detection on page 920 section, the TIMEOUT flag is set in the I2C_ISR register.
  
  Refer to Table 162: Examples of TIMEOUTB settings for various I2CCLK frequencies

**Caution:** Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

Bus Idle detection

In order to enable the tIDLE check, the 12-bit TIMEOUTA[11:0] field must be programmed with the timer reload value in order to obtain the tIDLE parameter. The TIDLE bit must be configured to ‘1’ in order to detect both SCL and SDA high level timeout.

Then the timer is enabled by setting the TIMOUTEN bit in the I2C_TIMEOUTR register. If both the SCL and SDA lines remain high for a time greater than (TIMEOUTA+1) x 4 x tI2CCLK, the TIMEOUT flag is set in the I2C_ISR register.

Refer to Table 163: Examples of TIMEOUTA settings for various I2CCLK frequencies (max tIDLE = 50 µs)
Caution: Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

31.4.13 SMBus: I2C_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to Section 31.3: I2C implementation.

- Configuring the maximum duration of \( t_{\text{TIMEOUT}} \) to 25 ms:

  Table 161. Examples of TIMEOUTA settings for various I2CCLK frequencies
  \( \text{(max } t_{\text{TIMEOUT}} = 25 \text{ ms)} \)

<table>
<thead>
<tr>
<th>( f_{\text{I2CCLK}} )</th>
<th>TIMEOUTA[11:0] bits</th>
<th>TIDLE bit</th>
<th>TIMEOUTEN bit</th>
<th>( t_{\text{TIMEOUT}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 MHz</td>
<td>0x61</td>
<td>0</td>
<td>1</td>
<td>98 x 2048 x 125 ns = 25 ms</td>
</tr>
<tr>
<td>16 MHz</td>
<td>0xC3</td>
<td>0</td>
<td>1</td>
<td>196 x 2048 x 62.5 ns = 25 ms</td>
</tr>
<tr>
<td>48 MHz</td>
<td>0x249</td>
<td>0</td>
<td>1</td>
<td>586 x 2048 x 20.08 ns = 25 ms</td>
</tr>
</tbody>
</table>

- Configuring the maximum duration of \( t_{\text{LOW}} \):SEXT and \( t_{\text{LOW}} \):MEXT to 8 ms:

  Table 162. Examples of TIMEOUTB settings for various I2CCLK frequencies

<table>
<thead>
<tr>
<th>( f_{\text{I2CCLK}} )</th>
<th>TIMEOUTB[11:0] bits</th>
<th>TEXTEN bit</th>
<th>( t_{\text{LOW}} ):EXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 MHz</td>
<td>0x1F</td>
<td>1</td>
<td>32 x 2048 x 125 ns = 8 ms</td>
</tr>
<tr>
<td>16 MHz</td>
<td>0x3F</td>
<td>1</td>
<td>64 x 2048 x 62.5 ns = 8 ms</td>
</tr>
<tr>
<td>48 MHz</td>
<td>0xBB</td>
<td>1</td>
<td>188 x 2048 x 20.08 ns = 8 ms</td>
</tr>
</tbody>
</table>

- Configuring the maximum duration of \( t_{\text{IDLE}} \) to 50 µs

  Table 163. Examples of TIMEOUTA settings for various I2CCLK frequencies
  \( \text{(max } t_{\text{IDLE}} = 50 \mu s) \)

<table>
<thead>
<tr>
<th>( f_{\text{I2CCLK}} )</th>
<th>TIMEOUTA[11:0] bits</th>
<th>TIDLE bit</th>
<th>TIMEOUTEN bit</th>
<th>( t_{\text{IDLE}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 MHz</td>
<td>0x63</td>
<td>1</td>
<td>1</td>
<td>100 x 4 x 125 ns = 50 µs</td>
</tr>
<tr>
<td>16 MHz</td>
<td>0xC7</td>
<td>1</td>
<td>1</td>
<td>200 x 4 x 62.5 ns = 50 µs</td>
</tr>
<tr>
<td>48 MHz</td>
<td>0x257</td>
<td>1</td>
<td>1</td>
<td>600 x 4 x 20.08 ns = 50 µs</td>
</tr>
</tbody>
</table>

31.4.14 SMBus slave mode

This section is relevant only when SMBus feature is supported. Refer to Section 31.3: I2C implementation.

In addition to I2C slave transfer management (refer to Section 31.4.8: I2C slave mode) some additional software flowcharts are provided to support SMBus.

SMBus Slave transmitter

When the IP is used in SMBus, SBC must be programmed to ‘1’ in order to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission.
that case the total number of TXIS interrupts is NBYTES-1 and the content of the I2C_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES-1 data transfer.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 309. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC**

- SMBus slave transmission
- Slave initialization
- \( \text{I2C_ISR.ADDR} < 1? \)
  - No
  - Read ADDCODE and DIR in I2C_ISR
    - \( \text{I2C_CR2.NBYTES} = N + 1 \)
    - PECBYTE=1
    - Set I2C_ICR.ADDRCF
  - Yes
- \( \text{I2C_ISR.TXIS} = 1? \)
  - No
  - Yes
- Write I2C_TXDR.TXDATA

SCL stretched
Figure 310. Transfer bus diagrams for SMBus slave transmitter (SBC=1)

Example SMBus slave transmitter 2 bytes + PEC,

<table>
<thead>
<tr>
<th>ADDR</th>
<th>TXIS</th>
<th>TXIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>EV1</td>
<td>data1</td>
<td>A</td>
</tr>
<tr>
<td>EV2</td>
<td>data2</td>
<td>P</td>
</tr>
<tr>
<td>EV3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NBYTES: 3

Legend:
- transmission
- reception
- SCL stretch

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

SMBus Slave receiver

When the I2C is used in SMBus mode, SBC must be programmed to ‘1’ in order to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD=1). Refer to Slave Byte Control mode on page 897 for more details.

In order to check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTES-1 data have been received, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

If no ACK software control is needed, the user can program PECBYTE=1 and, in the same write operation, program NBYTES with the number of bytes to be received in a continuous flow. After NBYTES-1 are received, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.
Figure 311. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC

- SMBus slave reception
  - Slave initialization
  - I2C_ISR.ADDR = 1?
    - No
    - Read ADDCODE and DIR in I2C_ISR
      - I2C_CR2.NBYTES = 1, RELOAD = 1
      - PECBYTE = 1
      - Set I2C_ICR.ADDRCF
    - Yes
      - I2C_ISR.RXNE = 1?
        - No
        - Read I2C_RXDR.RXDATA
          - Program I2C_CR2.NACK = 0
          - I2C_CR2.NBYTES = 1
          - N = N - 1
        - Yes
          - N = 1?
            - No
            - Read I2C_RXDR.RXDATA
              - Program RELOAD = 0
              - NACK = 0 and NBYTES = 1
            - Yes
              - I2C_ISR.RXNE = 1?
                - No
                - Read I2C_RXDR.RXDATA
    - Yes
      - I2C_ISR.TCR = 1?
        - No
        - SCL stretched
        - Read I2C_RXDR.RXDATA
          - Program I2C_RXDR.RXDATA
            - RELOAD = 0
            - NACK = 0 and NBYTES = 1
This section is relevant only when SMBus feature is supported. Refer to Section 31.3: I2C implementation.

In addition to I2C master transfer management (refer to Section 31.4.9: I2C master mode) some additional software flowcharts are provided to support SMBus.

**SMBus Master transmitter**

When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES+1. So if the PECBYTE bit is set when NBYTES=0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, automatic end mode must be selected (AUTOEND=1). In this case, the STOP condition automatically follows the PEC transmission.
When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND=0). In this case, once NBYTES-1 have been transmitted, the I2C_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 313. Bus transfer diagrams for SMBus master transmitter**

Example SMBus master transmitter 2 bytes + PEC, automatic end mode (STOP)

| INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECBYTE, set START |
| EV1: TXIS ISR: wr data1 |
| EV2: TXIS ISR: wr data2 |

Example SMBus master transmitter 2 bytes + PEC, software end mode (RESTART)

| INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECBYTE, set START |
| EV1: TXIS ISR: wr data1 |
| EV2: TXIS ISR: wr data2 |
| EV3: TC ISR: program Slave address, program NBYTES = N, set START |
SMBus Master receiver

When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND=1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND=0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.
Figure 314. Bus transfer diagrams for SMBus master receiver

Example SMBus master receiver 2 bytes + PEC, automatic end mode (STOP)

INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECBYTE, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2
EV3: RXNE ISR: rd PEC

Example SMBus master receiver 2 bytes + PEC, software end mode (RESTART)

INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECBYTE, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2
EV3: RXNE ISR: read PEC
EV4: TC ISR: program Slave address, program NBYTES = N, set START
31.4.15 **Wakeup from Stop mode on address match**

This section is relevant only when Wakeup from Stop mode feature is supported. Refer to Section 31.3: I2C implementation.

The I2C is able to wakeup the MCU from Stop mode (APB clock is off), when it is addressed. All addressing modes are supported.

Wakeup from Stop mode is enabled by setting the WUPEN bit in the I2C_CR1 register. The HSI16 oscillator must be selected as the clock source for I2CCLK in order to allow wakeup from Stop mode.

During Stop mode, the HSI16 is switched off. When a START is detected, the I2C interface switches the HSI16 on, and stretches SCL low until HSI16 is woken up.

HSI16 is then used for the address reception.

In case of an address match, the I2C stretches SCL low during MCU wakeup time. The stretch is released when ADDR flag is cleared by software, and the transfer goes on normally.

If the address does not match, the HSI16 is switched off again and the MCU is not woken up.

*Note:* If the I2C clock is the system clock, or if WUPEN = 0, the HSI16 is not switched on after a START is received.

*Caution:* Only an ADDR interrupt can wakeup the MCU. Therefore do not enter Stop mode when the I2C is performing a transfer as a master, or as an addressed slave after the ADDR flag is set. This can be managed by clearing SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

31.4.16 **Error conditions**

The following are the error conditions which may cause communication to fail.

**Bus error (BERR)**

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of 9 SCL clock pulses. A START or a STOP condition is detected when a SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (i.e not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.
Arbitration lost (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In master mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the master switches automatically to slave mode.
- In slave mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in slave mode when NOSTRETCH=1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF=1 and the first data byte should be sent. The content of the I2C_TXDR register is sent if TXE=0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Packet Error Checking Error (PECERR)

This section is relevant only when the SMBus feature is supported. Refer to Section 31.3: I2C implementation.

A PEC error is detected when the received PEC byte does not match with the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Timeout Error (TIMEOUT)

This section is relevant only when the SMBus feature is supported. Refer to Section 31.3: I2C implementation.

A timeout error occurs for any of these conditions:

- TIDLE=0 and SCL remained low for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a SMBus timeout.
- TIDLE=1 and both SDA and SCL remained high for the time defined in the TIMEOUTA [11:0] bits: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus t_LOW:MEXT parameter)
- Slave cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus t_LOW:SEXT parameter)
When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

**Alert (ALERT)**

This section is relevant only when the SMBus feature is supported. Refer to *Section 31.3: I2C implementation.*

The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN=1), the alert pin detection is enabled (ALERTEN=1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2C.CR1 register.

### 31.4.17 DMA requests

#### Transmission using DMA

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the I2C_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see *Section 9: Direct memory access controller (DMA) on page 241*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to Master transmitter on page 908.

- In slave mode:
  - With NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOSTRETCH=1, the DMA must be initialized before the address match event.

- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to SMBus Slave transmitter on page 922 and SMBus Master transmitter on page 926.

**Note:** *If DMA is used for transmission, the TXIE bit does not need to be enabled.*

#### Reception using DMA

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured using the DMA peripheral (refer to *Section 9: Direct memory access controller (DMA) on page 241*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the
DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

- In slave mode with NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see Section 31.3: I2C implementation): the PEC transfer is managed with the NBYTES counter. Refer to SMBus Slave receiver on page 924 and SMBus Master receiver on page 928.

**Note:** If DMA is used for reception, the RXIE bit does not need to be enabled.

### 31.4.18 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG_I2Cx_SMBUS_TIMEOUT configuration bits in the DBG module.

### 31.5 I2C low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. I2C interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop$^{(1)}$</td>
<td>The I2C registers content is kept. If WUPEN = 1 and I2C is clocked by an internal oscillator (HSI16): the address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. If WUPEN=0: the I2C must be disabled before entering Stop mode</td>
</tr>
<tr>
<td>Standby</td>
<td>The I2C peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

$^{(1)}$ The ADDR match event can wake-up the device from Stop mode only if the I2C instance supports the Wakeup from Stop mode feature. Refer to I2C implementation table.
31.6 **I2C interrupts**

The table below gives the list of I2C interrupt requests.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit the Sleep mode</th>
<th>Exit the Stop mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C_EV</td>
<td>Receive buffer not empty</td>
<td>RXNE</td>
<td>RXIE</td>
<td>Read I2C_RXDR register</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Transmit buffer interrupt status</td>
<td>TXIS</td>
<td>TXIE</td>
<td>Write I2C_TXDR register</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stop detection interrupt flag</td>
<td>STOPF</td>
<td>STOPIE</td>
<td>Write STOPCF=1</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transfer Complete Reload</td>
<td>TCR</td>
<td>TCIE</td>
<td>Write I2C_CR2 with NBYTES[7:0] ≠ 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transfer complete</td>
<td>TC</td>
<td></td>
<td>Write START=1 or STOP=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Address matched</td>
<td>ADDR</td>
<td>ADDRIE</td>
<td>Write ADDRCF=1</td>
<td>Yes(1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NACK reception</td>
<td>NACKF</td>
<td>NACKIE</td>
<td>Write NACKCF=1</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>I2C_ER</td>
<td>Bus error</td>
<td>BERR</td>
<td></td>
<td>Write BERRCF=1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Arbitration loss</td>
<td>ARLO</td>
<td></td>
<td>Write ARLOCF=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overrun/Underrun</td>
<td>OVR</td>
<td></td>
<td>Write OVRCF=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PEC error</td>
<td>PECERR</td>
<td>ERRIE</td>
<td>Write PECERRCF=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Timeout/tLOW error</td>
<td>TIMEOUT</td>
<td></td>
<td>Write TIMEOUTCF=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SMBus Alert</td>
<td>ALERT</td>
<td></td>
<td>Write ALERTCF=1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the Wakeup from Stop mode feature. Refer to Section 31.3: I2C implementation.

31.7 **I2C registers**

Refer to Section 1.2 on page 50 for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

31.7.1 **I2C control register 1 (I2C_CR1)**

Address offset: 0x00

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.
### Bits 31:24 Reserved, must be kept at reset value.

**Bit 23 PECEN**: PEC enable

- 0: PEC calculation disabled
- 1: PEC calculation enabled

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 31.3: I2C implementation.*

**Bit 22 ALERTEN**: SMBus alert enable

- **Device mode (SMBHEN=0):**
  - 0: Releases SMBA pin high and Alert Response Address Header disabled: 0001100x followed by NACK.
  - 1: Drives SMBA pin low and Alert Response Address Header enables: 0001100x followed by ACK.

- **Host mode (SMBHEN=1):**
  - 0: SMBus Alert pin (SMBA) not supported.
  - 1: SMBus Alert pin (SMBA) supported.

*Note: When ALERTEN=0, the SMBA pin can be used as a standard GPIO. If the SMBus feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 31.3: I2C implementation.*

**Bit 21 SMBDEN**: SMBus Device Default address enable

- 0: Device default address disabled. Address 0b1100001x is NACKed.
- 1: Device default address enabled. Address 0b1100001x is ACKed.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 31.3: I2C implementation.*

**Bit 20 SMBHEN**: SMBus Host address enable

- 0: Host address disabled. Address 0b0001000x is NACKed.
- 1: Host address enabled. Address 0b0001000x is ACKed.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 31.3: I2C implementation.*

**Bit 19 GCEN**: General call enable

- 0: General call disabled. Address 0b00000000 is NACKed.
- 1: General call enabled. Address 0b00000000 is ACKed.

**Bit 18 WUPEN**: Wakeup from Stop mode enable

- 0: Wakeup from Stop mode disable.
- 1: Wakeup from Stop mode enable.

*Note: If the Wakeup from Stop mode feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 31.3: I2C implementation.*

*Note: WUPEN can be set only when DNF = ‘0000’*
Bit 17  **NOSTRETCH**: Clock stretching disable
This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.
0: Clock stretching enabled
1: Clock stretching disabled
*Note*: This bit can only be programmed when the I2C is disabled (PE = 0).

Bit 16  **SBC**: Slave byte control
This bit is used to enable hardware byte control in slave mode.
0: Slave byte control disabled
1: Slave byte control enabled

Bit 15  **RXDMAEN**: DMA reception requests enable
0: DMA mode disabled for reception
1: DMA mode enabled for reception

Bit 14  **TXDMAEN**: DMA transmission requests enable
0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13  Reserved, must be kept at reset value.

Bit 12  **ANOFF**: Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled
*Note*: This bit can only be programmed when the I2C is disabled (PE = 0).

Bits 11:8  **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * tI2CCLK
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to 1 tI2CCLK
...1111: digital filter enabled and filtering capability up to 15 tI2CCLK
*Note*: If the analog filter is also enabled, the digital filter is added to the analog filter. This filter can only be programmed when the I2C is disabled (PE = 0).

Bit 7  **ERRIE**: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled
*Note*: Any of these errors generate an interrupt:
Arbitration Loss (ARLO)
Bus Error detection (BERR)
Overrun/Underrun (OVR)
Timeout detection (TIMEOUT)
PEC error detection (PECERR)
Alert pin event detection (ALERT)

Bit 6  **TCIE**: Transfer Complete interrupt enable
0: Transfer Complete interrupt disabled
1: Transfer Complete interrupt enabled
*Note*: Any of these events generate an interrupt:
Transfer Complete (TC)
Transfer Complete Reload (TCR)
Bit 5 STOPIE: Stop detection Interrupt enable
0: Stop detection (STOPF) interrupt disabled
1: Stop detection (STOPF) interrupt enabled

Bit 4 NACKIE: Not acknowledge received Interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 ADDRIE: Address match Interrupt enable (slave only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 RXIE: RX Interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 TXIE: TX Interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 PE: Peripheral enable
0: Peripheral disable
1: Peripheral enable

Note: When PE=0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.

31.7.2 I2C control register 2 (I2C_CR2)
Address offset: 0x04
Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.
Bits 31:27  Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte
   This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE=0.
   0: No PEC transfer.
   1: PEC transmission/reception is requested
   **Note:** Writing '0' to this bit has no effect.
   This bit has no effect when RELOAD is set.
   This bit has no effect is slave mode when SBC=0.
   If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 31.3: I2C implementation.

Bit 25 **AUTOEND**: Automatic end mode (master mode)
   This bit is set and cleared by software.
   0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.
   1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.
   **Note:** This bit has no effect in slave mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode
   This bit is set and cleared by software.
   0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).
   1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes
   The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with SBC=0.
   **Note:** Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (slave mode)
   The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE=0.
   0: an ACK is sent after current received byte.
   1: a NACK is sent after current received byte.
   **Note:** Writing '0' to this bit has no effect.
   This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.
   When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.
   When hardware PEC checking is enabled (PECBYTE=1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: Stop generation (master mode)
   The bit is set by software, cleared by hardware when a STOP condition is detected, or when PE = 0.
   **In Master Mode:**
   0: No Stop generation.
   1: Stop generation after current byte transfer.
   **Note:** Writing '0' to this bit has no effect.
Bit 13 **START**: Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software by writing ‘1’ to the ADDRCF bit in the I2C_ICR register.

- 0: No Start generation.
- 1: Restart/Start generation:
  - If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated Start condition when RELOAD=0, after the end of the NBYTES transfer.
  - Otherwise setting this bit generates a START condition once the bus is free.

*Note:* Writing ‘0’ to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in slave mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (master receiver mode)

- 0: The master sends the complete 10 bit slave address read sequence: Start + 2 bytes 10bit address in write direction + Restart + 1st 7 bits of the 10 bit address in read direction.
- 1: The master only sends the 1st 7 bits of the 10 bit address, followed by Read direction.

*Note:* Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (master mode)

- 0: The master operates in 7-bit addressing mode,
- 1: The master operates in 10-bit addressing mode

*Note:* Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (master mode)

- 0: Master requests a write transfer.
- 1: Master requests a read transfer.

*Note:* Changing this bit when the START bit is set is not allowed.

Bits 9:8 **SADD[9:8]**: Slave address bit 9:8 (master mode)

- In 7-bit addressing mode (ADD10 = 0):
  - These bits are don’t care
- In 10-bit addressing mode (ADD10 = 1):
  - These bits should be written with bits 9:8 of the slave address to be sent

*Note:* Changing these bits when the START bit is set is not allowed.

Bits 7:1 **SADD[7:1]**: Slave address bit 7:1 (master mode)

- In 7-bit addressing mode (ADD10 = 0):
  - These bits should be written with the 7-bit slave address to be sent
- In 10-bit addressing mode (ADD10 = 1):
  - These bits should be written with bits 7:1 of the slave address to be sent.

*Note:* Changing these bits when the START bit is set is not allowed.

Bit 0 **SADD0**: Slave address bit 0 (master mode)

- In 7-bit addressing mode (ADD10 = 0):
  - This bit is don’t care
- In 10-bit addressing mode (ADD10 = 1):
  - This bit should be written with bit 0 of the slave address to be sent

*Note:* Changing these bits when the START bit is set is not allowed.
31.7.3  I2C own address 1 register (I2C_OAR1)

Address offset: 0x08
Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

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

Bits 31:16  Reserved, must be kept at reset value.

Bit 15  OA1EN: Own Address 1 enable
          0: Own address 1 disabled. The received slave address OA1 is NACKed.
          1: Own address 1 enabled. The received slave address OA1 is ACKed.

Bits 14:11  Reserved, must be kept at reset value.

Bit 10  OA1MODE: Own Address 1 10-bit mode
          0: Own address 1 is a 7-bit address.
          1: Own address 1 is a 10-bit address.

*Note: This bit can be written only when OA1EN=0.*

Bits 9:8  OA1[9:8]: Interface address
          7-bit addressing mode: do not care
          10-bit addressing mode: bits 9:8 of address

*Note: These bits can be written only when OA1EN=0.*

Bits 7:1  OA1[7:1]: Interface address
          7-bit addressing mode: 7-bit address
          10-bit addressing mode: bits 7:1 of 10-bit address

*Note: These bits can be written only when OA1EN=0.*

Bit 0  OA1[0]: Interface address
          7-bit addressing mode: do not care
          10-bit addressing mode: bit 0 of address

*Note: This bit can be written only when OA1EN=0.*
### 31.7.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>OA2EN</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own Address 2 enable

0: Own address 2 disabled. The received slave address OA2 is NACKed.
1: Own address 2 enabled. The received slave address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own Address 2 masks

000: No mask
001: OA2[1] is masked and don’t care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don’t care. Only OA2[7:3] are compared.
100: OA2[4:1] are masked and don’t care. Only OA2[7:5] are compared.
111: OA2[7:1] are masked and don’t care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

*Note*: These bits can be written only when OA2EN=0.

As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address

7-bit addressing mode: 7-bit address

*Note*: These bits can be written only when OA2EN=0.

Bit 0 Reserved, must be kept at reset value.
31.7.5  I2C timing register (I2C_TIMINGR)

Address offset: 0x10
Reset value: 0x0000 0000
Access: No wait states

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

Bits 31:28  **PRESC[3:0]**: Timing prescaler
This field is used to prescale I2CCLK in order to generate the clock period t_{PRESC} used for data setup and hold counters (refer to I2C timings on page 889) and for SCL high and low level counters (refer to I2C master initialization on page 904).
\[ t_{PRESC} = (\text{PRESC} + 1) \times f_{I2CCLK} \]

Bits 27:24  Reserved, must be kept at reset value.

Bits 23:20  **SCLDEL[3:0]**: Data setup time
This field is used to generate a delay t_{SCLDEL} between SDA edge and SCL rising edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.
\[ t_{SCLDEL} = (\text{SCLDEL} + 1) \times t_{PRESC} \]
*Note:* \( t_{SCLDEL} \) is used to generate \( t_{SU:DAT} \) timing.

Bits 19:16  **SDADEL[3:0]**: Data hold time
This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.
\[ t_{SDADEL} = \text{SDADEL} \times t_{PRESC} \]
*Note:* \( t_{SDADEL} \) is used to generate \( t_{HD:DAT} \) timing.

Bits 15:8  **SCLH[7:0]**: SCL high period (master mode)
This field is used to generate the SCL high period in master mode.
\[ t_{SCLH} = (\text{SCLH} + 1) \times t_{PRESC} \]
*Note:* \( t_{SCLH} \) is also used to generate \( t_{SU:STO} \) and \( t_{HD:STA} \) timing.

Bits 7:0  **SCLL[7:0]**: SCL low period (master mode)
This field is used to generate the SCL low period in master mode.
\[ t_{SCLL} = (\text{SCLL} + 1) \times t_{PRESC} \]
*Note:* \( t_{SCLL} \) is also used to generate \( t_{BUF} \) and \( t_{SU:STA} \) timings.

*Note:* This register must be configured when the I2C is disabled (PE = 0).

*Note:* The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.
31.7.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14
Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to “0x00000000”. Refer to Section 31.3: I2C implementation.

| Bit 31  | TEXTEN   | 0: Extended clock timeout detection is disabled  
|        |          | 1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than \( t_{\text{LOW:EXT}} \) is done by the I2C interface, a timeout error is detected (TIMEOUT=1). |

Bits 30:28 Reserved, must be kept at reset value.

|           |               | This field is used to configure the cumulative clock extension timeout: |
|           |               | In master mode, the master cumulative clock low extend time (\( t_{\text{LOW:EXT}} \)) is detected |
|           |               | In slave mode, the slave cumulative clock low extend time (\( t_{\text{LOW:SEXT}} \)) is detected |
|           |               | \( t_{\text{LOW:EXT}} = (\text{TIMEOUTB}+1) \times 2048 \times t_{\text{I2CCLK}} \) |
|           |               | Note: These bits can be written only when TEXTEN=0. |

| Bit 15  | TIMOUTEN   | Clock timeout enable |
|        | 0: SCL timeout detection is disabled  
|        | 1: SCL timeout detection is enabled: when SCL is low for more than \( t_{\text{TIMEOUT}} \) (TIDLE=0) or high for more than \( t_{\text{IDLE}} \) (TIDLE=1), a timeout error is detected (TIMEOUT=1). |

Bits 14:13 Reserved, must be kept at reset value.

| Bit 12  | TIDLE   | Idle clock timeout detection |
|        | 0: TIMEOUTA is used to detect SCL low timeout  
|        | 1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition) |
|        | Note: This bit can be written only when TIMOUTEN=0. |

| Bits 11:0 | TIMEOUTA[11:0] | Bus Timeout A |
|          |               | This field is used to configure: |
|          |               | The SCL low timeout condition \( t_{\text{TIMEOUT}} \) when TIDLE=0 |
|          |               | \( t_{\text{TIMEOUT}} = (\text{TIMEOUTA}+1) \times 2048 \times t_{\text{I2CCLK}} \) |
|          |               | The bus idle condition (both SCL and SDA high) when TIDLE=1 |
|          |               | \( t_{\text{IDLE}} = (\text{TIMEOUTA}+1) \times 4 \times t_{\text{I2CCLK}} \) |
|          |               | Note: These bits can be written only when TIMOUTEN=0. |

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to “0x00000000”. Refer to Section 31.3: I2C implementation.
31.7.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18
Reset value: 0x0000 0001
Access: No wait states

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>-------------</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>BUSY</td>
<td>Res</td>
<td>ALERT</td>
<td>TIMEOUT</td>
<td>PEC</td>
<td>ERR</td>
<td>OVR</td>
<td>ARLO</td>
<td>BERR</td>
<td>TCR</td>
<td>TC</td>
<td>STOPF</td>
<td>NACKF</td>
<td>ADDR</td>
<td>RXNE</td>
<td>TXIS</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>------</td>
<td>-------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>----</td>
<td>----</td>
<td>------</td>
<td>------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>rs</td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]** Address match code (Slave mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1).
In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the 2 MSBs of the address.

Bit 16 **DIR**: Transfer direction (Slave mode)

This flag is updated when an address match event occurs (ADDR=1).
0: Write transfer, slave enters receiver mode.
1: Read transfer, slave enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected, or when PE=0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN=1 (SMBus host configuration), ALERTEN=1 and a SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

**Note**: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 31.3: I2C implementation.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

**Note**: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 31.3: I2C implementation.
Bit 11 **PECERR**: PEC Error in reception
This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

*Note: This bit is cleared by hardware when PE=0.*

If the SMBus feature is not supported, this bit is reserved and forced by hardware to ‘0’. Refer to Section 31.3: I2C implementation.

Bit 10 **OVR**: Overrun/Underrun (slave mode)
This flag is set by hardware in slave mode with NOSTRETCH=1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

*Note: This bit is cleared by hardware when PE=0.*

Bit 9 **ARLO**: Arbitration lost
This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

*Note: This bit is cleared by hardware when PE=0.*

Bit 8 **BERR**: Bus error
This flag is set by hardware when a misplaced Start or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting BERRCF bit.

*Note: This bit is cleared by hardware when PE=0.*

Bit 7 **TCR**: Transfer Complete Reload
This flag is set by hardware when RELOAD=1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

*Note: This bit is cleared by hardware when PE=0.*

This flag is only for master mode, or for slave mode when the SBC bit is set.

Bit 6 **TC**: Transfer Complete (master mode)
This flag is set by hardware when RELOAD=0, AUTOEND=0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

*Note: This bit is cleared by hardware when PE=0.*

Bit 5 **STOPF**: Stop detection flag
This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:
- either as a master, provided that the STOP condition is generated by the peripheral.
- or as a slave, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

*Note: This bit is cleared by hardware when PE=0.*

Bit 4 **NACKF**: Not Acknowledge received flag
This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

*Note: This bit is cleared by hardware when PE=0.*

Bit 3 **ADDR**: Address matched (slave mode)
This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting ADDRCF bit.

*Note: This bit is cleared by hardware when PE=0.*
31.7.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C
Reset value: 0x0000 0000
Access: No wait states

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

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 ALERTCF: Alert flag clear

Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 31.3: I2C implementation.

Bit 12 TIMOUTCF: Timeout detection flag clear

Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 31.3: I2C implementation.

Bit 11 PECCF: PEC Error flag clear

Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 31.3: I2C implementation.
Bit 10 **OVRCF**: Overrun/Underrun flag clear  
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear  
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear  
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear  
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not Acknowledge flag clear  
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear  
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

### 31.7.9 I2C PEC register (I2C_PECR)

Address offset: 0x20  
Reset value: 0x0000 0000  
Access: No wait states

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

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

PEC[7:0]  
```

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

Bits 7:0 **PEC[7:0]** Packet error checking register  
This field contains the internal PEC when PECEN=1.  
The PEC is cleared by hardware when PE=0.

**Note:** If the SMBus feature is not supported, this register is reserved and forced by hardware to “0x00000000”. Refer to Section 31.3: I2C implementation.
31.7.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24
Reset value: 0x0000 0000
Access: No wait states

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

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

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

Bits 7:0 RXDATA[7:0] 8-bit receive data
Data byte received from the I2C bus

31.7.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28
Reset value: 0x0000 0000
Access: No wait states

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

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

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

Bits 7:0 TXDATA[7:0] 8-bit transmit data
Data byte to be transmitted to the I2C bus

Note: These bits can be written only when TXE=1.
# I2C register map

The table below provides the I2C register map and reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>I2C_CR1</td>
<td></td>
<td>0x4</td>
<td>I2C_CR2</td>
<td></td>
<td>0x8</td>
<td>I2C_OAR1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>I2C_TIMINGR</td>
<td></td>
<td>0x14</td>
<td>I2C_TIMEOUTR</td>
<td></td>
<td>0x18</td>
<td>I2C_ISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>I2C_ICR</td>
<td></td>
<td>0x20</td>
<td>I2C_PECR</td>
<td></td>
<td>0x24</td>
<td>I2C_RXDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 166. I2C register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>I2C_CR1</td>
<td></td>
<td>0x4</td>
<td>I2C_CR2</td>
<td></td>
<td>0x8</td>
<td>I2C_OAR1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>I2C_TIMINGR</td>
<td></td>
<td>0x14</td>
<td>I2C_TIMEOUTR</td>
<td></td>
<td>0x18</td>
<td>I2C_ISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>I2C_ICR</td>
<td></td>
<td>0x20</td>
<td>I2C_PECR</td>
<td></td>
<td>0x24</td>
<td>I2C_RXDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>I2C_CR1</td>
<td></td>
<td>0x4</td>
<td>I2C_CR2</td>
<td></td>
<td>0x8</td>
<td>I2C_OAR1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>I2C_TIMINGR</td>
<td></td>
<td>0x14</td>
<td>I2C_TIMEOUTR</td>
<td></td>
<td>0x18</td>
<td>I2C_ISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>I2C_ICR</td>
<td></td>
<td>0x20</td>
<td>I2C_PECR</td>
<td></td>
<td>0x24</td>
<td>I2C_RXDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>I2C_CR1</td>
<td></td>
<td>0x4</td>
<td>I2C_CR2</td>
<td></td>
<td>0x8</td>
<td>I2C_OAR1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>I2C_TIMINGR</td>
<td></td>
<td>0x14</td>
<td>I2C_TIMEOUTR</td>
<td></td>
<td>0x18</td>
<td>I2C_ISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>I2C_ICR</td>
<td></td>
<td>0x20</td>
<td>I2C_PECR</td>
<td></td>
<td>0x24</td>
<td>I2C_RXDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>I2C_CR1</td>
<td></td>
<td>0x4</td>
<td>I2C_CR2</td>
<td></td>
<td>0x8</td>
<td>I2C_OAR1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>I2C_TIMINGR</td>
<td></td>
<td>0x14</td>
<td>I2C_TIMEOUTR</td>
<td></td>
<td>0x18</td>
<td>I2C_ISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>I2C_ICR</td>
<td></td>
<td>0x20</td>
<td>I2C_PECR</td>
<td></td>
<td>0x24</td>
<td>I2C_RXDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>I2C_CR1</td>
<td></td>
<td>0x4</td>
<td>I2C_CR2</td>
<td></td>
<td>0x8</td>
<td>I2C_OAR1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>I2C_TIMINGR</td>
<td></td>
<td>0x14</td>
<td>I2C_TIMEOUTR</td>
<td></td>
<td>0x18</td>
<td>I2C_ISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>I2C_ICR</td>
<td></td>
<td>0x20</td>
<td>I2C_PECR</td>
<td></td>
<td>0x24</td>
<td>I2C_RXDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>I2C_CR1</td>
<td></td>
<td>0x4</td>
<td>I2C_CR2</td>
<td></td>
<td>0x8</td>
<td>I2C_OAR1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>I2C_TIMINGR</td>
<td></td>
<td>0x14</td>
<td>I2C_TIMEOUTR</td>
<td></td>
<td>0x18</td>
<td>I2C_ISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>I2C_ICR</td>
<td></td>
<td>0x20</td>
<td>I2C_PECR</td>
<td></td>
<td>0x24</td>
<td>I2C_RXDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Refer to Section 2.2 on page 54 for the register boundary addresses.

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x28   | I2C_TXDR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
32 Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

32.1 USART introduction

The USART offers a flexible means to perform Full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and Half-duplex Single-wire communications, as well as LIN (local interconnection network), Smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and Modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.
### 32.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data
  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wakeup from Mute mode by idle line detection or address mark detection
## 32.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T=0 and T=1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for Smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

## 32.4 USART implementation

*Table 167* describes USART implementation on STM32G0x1 devices. It also includes LPUART for comparison.

<table>
<thead>
<tr>
<th>USART modes/features(^{(1)})</th>
<th>USART1(^{(2)})</th>
<th>USART2(^{(2)})</th>
<th>USART3/4(^{(2)})</th>
<th>LPUART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware flow control for modem</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Continuous communication using DMA</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Multiprocessor communication</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Synchronous mode (Master/Slave)</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Smartcard mode</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Single-wire Half-duplex communication</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IrDA SIR ENDEC block</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LIN mode</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Dual clock domain and wakeup from low-power mode</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Receiver timeout interrupt</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Modbus communication</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Auto baud rate detection</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Driver Enable</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>USART data length</td>
<td>7, 8 and 9 bits</td>
<td>7, 8 and 9 bits</td>
<td>7, 8 and 9 bits</td>
<td>7, 8 and 9 bits</td>
</tr>
<tr>
<td>Tx/Rx FIFO</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Tx/Rx FIFO size</td>
<td>8</td>
<td>-</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

1. X = supported.
2. Only for STM32G071xx and STM32G081xx
3. Only for STM32G031xx and STM32G041xx
32.5 USART functional description

32.5.1 USART block diagram

The simplified block diagram given in Figure 315 shows two fully-independent clock domains:

- The **usart_pclk** clock domain
  
  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.

- The **usart_ker_ck** kernel clock domain.
  
  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is disabled, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external SCLK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.
32.5.2 USART signals

USART bidirectional communications

USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX (Receive Data Input)**
  RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

- **TX (Transmit Data Output)**
  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In Single-wire and Smartcard modes, this I/O is used to transmit and receive data.

RS232 Hardware flow control mode

The following pins are required in RS232 Hardware flow control mode:

- **CTS (Clear To Send)**
  When driven high, this signal blocks the data transmission at the end of the current transfer.

- **RTS (Request To Send)**
  When it is low, this signal indicates that the USART is ready to receive data.

RS485 Hardware control mode

The following pin is required in RS485 Hardware control mode:

- **DE (Driver Enable)**
  This signal activates the transmission mode of the external transceiver.

*Note: DE and RTS share the same pin.*

Synchronous master/slave mode and Smartcard mode

The following pin is required in synchronous master/slave mode and Smartcard mode:

- **CK**
  This pin acts as Clock output in Synchronous master and Smartcard modes. It acts as Clock input is Synchronous slave mode.
  In Synchronous Master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable.
  In Smartcard mode, CK output provides the clock to the smartcard.

- **NSS**
  This pin acts as Slave Select input in Synchronous slave mode.

*Note: NSS and CTS share the same pin.*
32.5.3 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see Figure 316):

- 7-bit character length: M[1:0] = ‘10’
- 8-bit character length: M[1:0] = ‘00’
- 9-bit character length: M[1:0] = ‘01’

Note: In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

An Idle character is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A Break character is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.
Figure 316. Word length programming

9-bit word length (M = 01), 1 Stop bit

Data frame

Start bit     Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Bit8 Stop bit

Possible Parity bit

Next Start bit

Idle frame

Break frame

8-bit word length (M = 00), 1 Stop bit

Data frame

Start bit     Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Stop bit

Possible Parity bit

Next Start bit

Idle frame

Break frame

7-bit word length (M = 10), 1 Stop bit

Data frame

Start bit     Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Stop bit

Possible Parity bit

Next Start bit

Idle frame

Break frame

** LBCL bit controls last data clock pulse

MS33194V2
32.5.4 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and Smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

*Note:* The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in USART_CR3 control register.

In this case:

- The RXFT flag is set in the USART_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.
  
  This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.

  RXFTCFG data have been received: one data in USART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to ‘101’, the RXFT flag will be set when a number of data corresponding to the FIFO size has been received (FIFO size -1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data will not set the overrun flag.

- The TXFT flag is set in the USART_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.
  
  This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

32.5.5 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the SCLK pin.

**Character transmission**

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.
Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

**Note:** The TE bit must be set before writing the data to be transmitted to the USART_TDR. The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost. An idle frame is sent when the TE bit is enabled.

**Configurable stop bits**

The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- **1 stop bit:** This is the default value of number of stop bits.
- **2 stop bits:** This will be supported by normal USART, Single-wire and Modem modes.
- **1.5 stop bits:** To be used in Smartcard mode.

An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits (when M[1:0] = ’00’) or 11 low bits (when M[1:0] = ’01’) or 9 low bits (when M[1:0] = ’10’) followed by 2 stop bits (see **Figure 317**). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

![Figure 317. Configurable stop bits](image-url)
Character transmission procedure

To transmit a character, follow the sequence below:
1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in Section 32.5.10: USART multiprocessor communication.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC=1.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.
Single byte communication

- When FIFO mode is disabled
  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:
  - the data have been moved from the USART_TDR register to the shift register and the data transmission has started;
  - the USART_TDR register is empty;
  - the next data can be written to the USART_TDR register without overwriting the previous data.
  This flag generates an interrupt if the TXEIE bit is set.
  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.
  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  - the TXFIFO is not full;
  - the USART_TDR register is empty;
  - the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.
  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.
  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.
  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the microcontroller to enter the low-power mode (see Figure 318: TC/TXE behavior when transmitting).
Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see Figure 316).

If a ‘1’ is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIF0 is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

32.5.6 USART receiver

The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0 X 0 X 0 X 0.
Note: If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE=1, or RXFNE flag set and interrupt generated if RXFNEIE=1 if FIFO mode enabled) if the 3 sampled bits are at ‘0’ (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at ‘0’ and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at ‘0’).

The start bit is validated but the NE noise flag is set if,

a) for both samplings, 2 out of the 3 sampled bits are at ‘0’ (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)

or

b) for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at ‘0’.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).
Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to ‘1’.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in Section 32.5.10: USART multiprocessor communication.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty i.e. when there are data to be read from the RXFIFO.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to ‘1’ in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to ‘1’ in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be
generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

**Break character**

When a break character is received, the USART handles it as a framing error.

**Idle character**

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

**Overrun error**

- **FIFO mode disabled**
  
  An overrun error occurs if a character is received and RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte reception.
  
  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
  - the ORE bit is set;
  - the RDR content will not be lost. The previous data is available by reading the USART_RDR register.
  - the shift register will be overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXNEIE or the EIE bit is set.

- **FIFO mode enabled**
  
  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.
  
  Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.
  
  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:
  - The ORE bit is set.
  - The first entry in the RXFIFO will not be lost. It is available by reading the USART_RDR register.
  - The shift register will be overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART_ICR register.
Note: The ORE bit, when set, indicates that at least 1 data has been lost.
When the FIFO mode is disabled, there are two possibilities
- if RXNE=1, then the last valid data is stored in the receive register (RDR) and can be read,
- if RXNE=0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.

Selecting the clock source and the appropriate oversampling method
The choice of the clock source is done through the Clock Control system (see ). The clock source must be selected through the UE bit before enabling the USART.
The clock source must be selected according to two criteria:
- Possible use of the USART in low-power mode
- Communication speed.
The clock source frequency is usart_ker_ck.
When the dual clock domain and the wakeup from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see ). Otherwise the usart_ker_ck clock is the same as usart_pclk.
The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRESC register.

Figure 320. usart_ker_ck clock divider block diagram

Some usart_ker_ck sources allow the USART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.
For the other clock sources, the system must be active to allow USART communications.
The communication speed range (specially the maximum communication speed) is also determined by the clock source.
The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This allows obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.
The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see Figure 321 and Figure 322).
Depending on your application:

- select oversampling by 8 (OVER8=1) to achieve higher speed (up to \text{usart\_ker\_ck\_pres}/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to Section 32.5.8: Tolerance of the USART receiver to clock deviation on page 971).
- select oversampling by 16 (OVER8=0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum \text{usart\_ker\_ck\_pres}/16 (where \text{usart\_ker\_ck\_pres} is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART\_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

Depending on your application:

- select the three sample majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to Figure 168) because this indicates that a glitch occurred during the sampling.
- select the single sample method (ONEBIT=1) when the line is noise-free to increase the receiver tolerance to clock deviations (see Section 32.5.8: Tolerance of the USART receiver to clock deviation on page 971). In this case the NE bit will never be set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART\_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART\_CR3 register.

The NE bit is reset by setting NECF bit in USART\_ICR register.

Note: Noise error is not supported in SPI mode. Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to '0' by hardware.
**Figure 321. Data sampling when oversampling by 16**

**Figure 322. Data sampling when oversampling by 8**

**Table 168. Noise detection from sampled data**

<table>
<thead>
<tr>
<th>Sampled value</th>
<th>NE status</th>
<th>Received bit value</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing ‘1’ to the FECF in the USART_ICR register.

*Note:* Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.

- **0.5 stop bit (reception in Smartcard mode):** no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit:** sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **1.5 stop bits (Smartcard mode)**

  When transmitting in Smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE =1 in USART_CR1) and the stop bit is checked to test if the Smartcard has detected a parity error.

  In the event of a parity error, the Smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to Section 32.5.16: USART receiver timeout on page 984 for more details).

- **2 stop bits**

  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. The framing error flag is set if a framing error is detected during the first stop bit. The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.
32.5.7 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = ‘0’ or ‘1’)

In case of oversampling by 16, the baud rate is given by the following formula:

\[ \text{Tx/Rx baud} = \frac{\text{uart_ker_ckpres}}{\text{USARTDIV}} \]

In case of oversampling by 8, the baud rate is given by the following formula:

\[ \text{Tx/Rx baud} = \frac{2 \times \text{uart_ker_ckpres}}{\text{USARTDIV}} \]

Equation 2: baud rate in Smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

\[ \text{Tx/Rx baud} = \frac{\text{uart_ker_ckpres}}{\text{USARTDIV}} \]

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value should not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 baud with uart_ker_ck_pres = 8 MHz:

- In case of oversampling by 16:
  \[ \text{USARTDIV} = 8 \times 1000 \times 1000 / 9600 = 8333.33 \]
  \[ \text{BRR} = \text{USARTDIV} = 833 = 0341h \]
- In case of oversampling by 8:
  \[ \text{USARTDIV} = 2 \times 8 \times 1000 \times 1000 / 9600 = 1666.67 \]
  \[ \text{USARTDIV} = 1666,66 (1667d = 683h) \]
  \[ \text{BRR}[3:0] = 3h >>1 = 1h \]
  \[ \text{BRR} = 0x681 \]
Example 2
To obtain 921.6 Kbaud with usart_ker_ck_pres = 48 MHz:
- In case of oversampling by 16:
  \[ \text{USARTDIV} = \frac{48\,000\,000}{921\,600} \]
  \[ \text{BRR} = \text{USARTDIV} = 52d = 34h \]
- In case of oversampling by 8:
  \[ \text{USARTDIV} = 2 \times \frac{48\,000\,000}{921\,600} \]
  \[ \text{USARTDIV} = 104 (104d = 68h) \]
  \[ \text{BRR}[3:0] = \text{USARTDIV}[3:0] >> 1 = 8h >> 1 = 4h \]
  \[ \text{BRR} = 0x64 \]

32.5.8 Tolerance of the USART receiver to clock deviation
The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:
- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter’s local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

\[ \text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} + \text{DWU} < \text{USART receiver tolerance} \]

where
- \( \text{DWU} \) is the error due to sampling point deviation when the wakeup from low-power mode is used.
- \( \text{tWUUSART} \) is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

\[ \text{DWU} = \frac{t_{\text{WUUSART}}}{11 \times \text{Tbit}} \]

when \( \text{M}[1:0] = 01 \):

\[ \text{DWU} = \frac{t_{\text{WUUSART}}}{10 \times \text{Tbit}} \]

when \( \text{M}[1:0] = 00 \):

\[ \text{DWU} = \frac{t_{\text{WUUSART}}}{9 \times \text{Tbit}} \]

when \( \text{M}[1:0] = 10 \):
The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in Table 169, Table 170, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 169. Tolerance of the USART receiver when BRR[3:0] = 0000

<table>
<thead>
<tr>
<th>M bits</th>
<th>OVER8 bit = 0</th>
<th>OVER8 bit = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ONEBIT=0</td>
<td>ONEBIT=1</td>
</tr>
<tr>
<td>00</td>
<td>3.75%</td>
<td>4.375%</td>
</tr>
<tr>
<td>01</td>
<td>3.41%</td>
<td>3.97%</td>
</tr>
<tr>
<td>10</td>
<td>4.16%</td>
<td>4.86%</td>
</tr>
</tbody>
</table>

Table 170. Tolerance of the USART receiver when BRR[3:0] is different from 0000

<table>
<thead>
<tr>
<th>M bits</th>
<th>OVER8 bit = 0</th>
<th>OVER8 bit = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ONEBIT=0</td>
<td>ONEBIT=1</td>
</tr>
<tr>
<td>00</td>
<td>3.33%</td>
<td>3.88%</td>
</tr>
<tr>
<td>01</td>
<td>3.03%</td>
<td>3.53%</td>
</tr>
<tr>
<td>10</td>
<td>3.7%</td>
<td>4.31%</td>
</tr>
</tbody>
</table>

Note: The data specified in Table 169 and Table 170 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9-bit times when M = 10).

32.5.9 USART Auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism allows the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from USART_KER_CK_PRES/65535 and USART_KER_CK_PRES/16.
- When oversampling by 8, the baud rate ranges from USART_KER_CK_PRES/65535 and USART_KER_CK_PRES/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is
measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0**: Any character starting with a bit at ‘1’.
  In this case the USART measures the duration of the start bit (falling edge to rising edge).

- **Mode 1**: Any character starting with a 10xx bit pattern.
  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.

- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).
  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.

- **Mode 3**: A 0x55 character frame.
  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART will then wait for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag will be set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a ‘0’).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection should be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

*Note:* The BRR value might be corrupted if the USART is disabled (UE=0) during an auto baud rate operation.
32.5.10 **USART multiprocessor communication**

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the USART_CR1 register.

*Note:* *When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise Mute mode might remain active.*

When the Mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to ‘1’. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

**Idle line detection (WAKE=0)**

The USART enters Mute mode when the MMRQ bit is written to ‘1’ and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of Mute mode behavior using Idle line detection is given in *Figure 323.*
Figure 323. Mute mode using Idle line detection

Note: If the MMRQ is set while the IDLE character has already elapsed, Mute mode will not be entered (RWU is not set).

If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is set for this address byte and no interrupt or DMA request is issued when the USART enters Mute mode. When FIFO management is enabled, the software should ensure that there is at least one empty location in the RXFIFO before entering Mute mode.

The USART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in Mute mode

An example of Mute mode behavior using address mark detection is given in Figure 324.
32.5.11 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a Half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

**Modbus/RTU**

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the UART_CR2 register and the RTOIE in the UART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

**Modbus/ASCII**

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE=1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.
32.5.12 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in Table 171.

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit will be 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit will be 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).
32.5.13 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to Section 32.4: USART implementation on page 953.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:
- CLKEN in the USART_CR2 register,
- STOP[1:0], SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in Section 32.5.4 has to be applied for LIN Master transmission. It must be the same as for normal USART transmission with the following differences:
- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 ‘0’ bits as a break character. Then 2 bits of value ‘1’ are sent to allow the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE=1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL=1 in USART_CR2) consecutive bits are detected as ‘0’, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1’ is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at ‘0’, which will be the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1’, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the Figure 325: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 979.

Examples of break frames are given on Figure 326: Break detection in LIN mode vs. Framing error detection on page 980.
Figure 325. Break detection in LIN mode (11-bit break length - LBDL bit is set)

Case 1: break signal not long enough => break discarded, LBDF is not set

RX line
Capture strobe
Break state machine
Break frame

<table>
<thead>
<tr>
<th>Bit0</th>
<th>Bit1</th>
<th>Bit2</th>
<th>Bit3</th>
<th>Bit4</th>
<th>Bit5</th>
<th>Bit6</th>
<th>Bit7</th>
<th>Bit8</th>
<th>Bit9</th>
<th>Bit10</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Read samples: 0 0 0 0 0 0 0 0 0 1

Case 2: break signal just long enough => break detected, LBDF is set

RX line
Capture strobe
Break state machine
Break frame
Delimiter is immediate

<table>
<thead>
<tr>
<th>Bit0</th>
<th>Bit1</th>
<th>Bit2</th>
<th>Bit3</th>
<th>Bit4</th>
<th>Bit5</th>
<th>Bit6</th>
<th>Bit7</th>
<th>Bit8</th>
<th>Bit9</th>
<th>Bit10</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Read samples: 0 0 0 0 0 0 0 0 0 0

Case 3: break signal long enough => break detected, LBDF is set

RX line
Capture strobe
Break state machine
Break frame
wait delimiter

<table>
<thead>
<tr>
<th>Bit0</th>
<th>Bit1</th>
<th>Bit2</th>
<th>Bit3</th>
<th>Bit4</th>
<th>Bit5</th>
<th>Bit6</th>
<th>Bit7</th>
<th>Bit8</th>
<th>Bit9</th>
<th>Bit10</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Read samples: 0 0 0 0 0 0 0 0 0 0

LBDF
32.5.14 USART synchronous mode

Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to ‘1’. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The SCLK pin is the output of the USART transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see Figure 327, Figure 328 and Figure 329).

During the Idle state, preamble and send break, the external SCLK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since SCLK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on SCLK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).
Note: In master mode, the SCLK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 327. USART example of synchronous master transmission

![ USART example of synchronous master transmission diagram ]

Figure 328. USART data clock timing diagram in synchronous master mode (M bits =00)

![ USART data clock timing diagram in synchronous master mode diagram ]
Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to ‘1’. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The SCLK pin is the input of the USART in slave mode.

Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (usart_ker_ck_pres) must be greater than 3 times the CK input frequency.

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see Figure 330).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.
Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- **Software NSS management (DIS_NSS = 1)**
  
  SPI slave will always be selected and NSS input pin will be ignored.
  
  The external NSS pin remains free for other application uses.

- **Hardware NSS management (DIS_NSS = 0)**
  
  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

**Note:** The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE=0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it will become desynchronized with the master.

The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave will transmit zeros.

**SPI Slave underrun error**

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.
In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error will allow sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

*Note:* An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first SCLK transmission edge. To avoid this underrun error, the USART_TDR should be written 3 $\text{usart}_k\text{er}_c\text{k}$ cycles before the first SCLK edge.

### 32.5.15 USART single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to ‘1’:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

### 32.5.16 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = ‘00’ or STOP = ‘11’
- from the end of the second stop bit if STOP = ‘10’.
- from the beginning of the stop bit if STOP = ‘01’.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout will be generated if RTOIE bit in USART_CR1 register is set.
32.5.17 USART Smartcard mode

This section is relevant only when Smartcard mode is supported. Refer to Section 32.4: USART implementation on page 953.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In Smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the Smartcard.

The Smartcard interface is designed to support asynchronous Smartcard protocol as defined in the ISO 7816-3 standard. Both T=0 (character mode) and T=1 (block mode) are supported.

The USART should be configured as:

- 8 bits plus parity: M=1 and PCE=1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP='11' in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T=0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

*Figure 331* shows examples of what can be seen on the data line with and without parity error.

![Figure 331. ISO 7816-3 asynchronous protocol](image)

When connected to a Smartcard, the TX output of the USART drives a bidirectional line that is also driven by the Smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the Smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.
The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

- **Smartcard auto-retry in transmission:** A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.

- **If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T=1 mode).** If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the Smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.

- **Smartcard auto-retry in reception:** the BUSY flag remains set if the USART NACKs the card but the card doesn’t repeat the character.

- **In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).**

- **The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.**

- **The de-assertion of TC flag is unaffected by Smartcard mode.**

- **If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.**

- **On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.**

**Note:** Break characters are not significant in Smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

*No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.*

**Figure 332** shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.
The USART can provide a clock to the Smartcard through the SCLK output. In Smartcard mode, SCLK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. SCLK frequency can be programmed from \( \text{usart_ker_ck_pres}/2 \) to \( \text{usart_ker_ck_pres}/62 \), where \( \text{usart_ker_ck_pres} \) is the peripheral input clock divided by a programmed prescaler.

**Block mode (T=1)**

In \( T=1 \) (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the Smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt will be generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the Smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time -11 value), in order to allow the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the Smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART will signal it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: As in the Smartcard protocol definition, the BWT/CWT values should be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT - 11 or CWT - 11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the Smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value.
With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value will be programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

**Note:** The error checking code (LRC/CRC) must be computed/verified by software.

### Direct and inverse convention

The Smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=0, DATAINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=1, DATAINV=1.

**Note:** When logical data values are inverted (0=H, 1=L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to ‘3F’.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to ‘3B’.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH => the USART received character will be ‘03’ and the parity will be odd.
Therefore, two methods are available for TS pattern recognition:

**Method 1**

The USART is programmed in standard Smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention.
- In response to the error signal, the card retries the same TS character, and it will be correctly received this time, by the reprogrammed USART.

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

**Method 2**

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103 -> inverse convention to be chosen
(H) LHHL HHH LLH = 0x13B -> direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

### 32.5.18 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to Section 32.4: USART implementation on page 953.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see Figure 333).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 Kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not
encoded. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.

- A ‘0’ is transmitted as a high pulse and a ‘1’ is transmitted as a ‘0’. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see Figure 334).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods will be accepted as a pulse. The IrDA encoder/decoder doesn’t work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to ‘1 stop bit’.

**IrDA low-power mode**

- **Transmitter**
  In low-power mode, the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < PSC< 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.

- **Receiver**
  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in the USART_GTPR).

*Note:* A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.

The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).
Figure 333. IrDA SIR ENDEC block diagram

Figure 334. IrDA data modulation (3/16) - Normal mode
32.5.19 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 32.4: USART implementation on page 953 to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 32.5.6. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding Direct memory access controller section) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.
Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to the corresponding Direct memory access controller section) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register.
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

32.5.20 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The Figure 337 shows how to connect 2 devices in this mode:
RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to ‘1’ in the USART_CR3 register.

**RS232 RTS flow control**

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. Figure 338 shows an example of communication with RTS flow control enabled.

![Figure 338. RS232 RTS flow control](image)

*Note:* When FIFO mode is enabled, nRTS is de-asserted only when RXFIFO is full.

**RS232 CTS flow control**

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. Figure 339 shows an example of communication with CTS flow control enabled.
Figure 339. RS232 CTS flow control

Note: For correct behavior, nCTS must be asserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This allows the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the USART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).
32.5.21 **USART low-power management**

The USART has advanced low-power mode functions, that allow transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the USART provides a wakeup interrupt (usart_wkup) if a specific action requiring the activation of the usart_pclk clock is needed:

- **If FIFO mode is disabled**
  - usart_pclk clock has to be activated to empty the USART data register.
  - In this case, the usart_wkup interrupt source is RXNE set to ‘1’. The RXNEIE bit must be set before entering low-power mode.

- **If FIFO mode is enabled**
  - usart_pclk clock has to be activated to:
    - to fill the TXFIFO
    - or to empty the RXFIFO
  - In this case, the usart_wkup interrupt source can be:
    - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
    - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
    - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

This allows sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart_wkup interrupt source can be one of the following events:

- TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
- RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

For example, the application can set the threshold to the maximum RXFIFO size if the wakeup time is less than the time required to receive a single byte across the line.

Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wakeup the MCU from low-power mode allows doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific usart_wkup interrupt can be selected through the WUS bitfields.

When the wakeup event is detected, the WUF flag is set by hardware and a usart_wkup interrupt is generated if the WUFIE bit is set. In this case the usart_wkup interrupt is not mandatory and setting the WUF being is sufficient to wake up the MCU from low-power mode.
Note: Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in low-power or active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled when exiting from low-power mode.

When the FIFO is enabled, waking up from low-power mode on address match is only possible when Mute mode is enabled.

Using Mute mode with low-power mode

If the USART is put into Mute mode before entering low-power mode:

• Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.

• If the wakeup from Mute mode on address match is used, then the low-power mode wakeup source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface will remain in Mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, Mute mode can be used with wakeup from low-power mode without any constraints (i.e. the two points mentioned above about Mute and low-power mode are valid only when FIFO management is disabled).

Wakeup from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.

If the wakeup event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wakeup event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wakeup event programmed to “address match detection” and FIFO management disabled.
Figure 340 shows the USART behavior when the wakeup event is verified.

**Figure 340. Wakeup event verified (wakeup event = address match, FIFO disabled)**

![Figure 340](MSv40856V2)

Note: The figures above are valid when address match or any received frame is used as wakeup event. If the wakeup event is the start bit detection, the USART sends the wakeup event to the MCU at the end of the start bit.

Figure 341 shows the USART behavior when the wakeup event is not verified.

**Figure 341. Wakeup event not verified (wakeup event = address match, FIFO disabled)**

![Figure 341](MSv40857V2)
Determining the maximum USART baud rate that allows to correctly wake up the microcontroller from low-power mode

The maximum baud rate allowing to correctly wake up the microcontroller from low-power mode depends on the wakeup time parameter (refer to the device datasheet) and on the USART receiver tolerance (see Section 32.5.8: Tolerance of the USART receiver to clock deviation).

Let us take the example of OVER8 = 0, M bits = ‘01’, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to Table 169: Tolerance of the USART receiver when BRR [3:0] = 0000, the USART receiver tolerance equals 3.41%.

\[ D_{WU\text{max}} = t_{WU\text{USART}} / (11 \times T_{\text{bit Min}}) \]
\[ T_{\text{bit Min}} = t_{WU\text{USART}} / (11 \times D_{WU\text{max}}) \]
where \( t_{WU\text{USART}} \) is the wakeup time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the \( \text{usart\_ker\_ck} \) inaccuracy.

For example, if HSI is used as \( \text{usart\_ker\_ck} \), and the HSI inaccuracy is of 1%, then we obtain:

\[ t_{WU\text{USART}} = 3 \mu s \text{ (values provided only as examples; for correct values, refer to the device datasheet).} \]
\[ D_{WU\text{max}} = 3.41\% - 1\% = 2.41\% \]
\[ T_{\text{bit min}} = 3 \mu s / (11 \times 2.41\%) = 11.32 \mu s. \]

As a result, the maximum baud rate allowing to wakeup correctly from low-power mode is: \( 1/11.32 \mu s = 88.36 \) Kbaud.

32.6 USART interrupts

During USART communications, an interrupt (\( \text{usart\_it} \)) can be generated by different events. The USART block can also generate a wakeup interrupt (\( \text{usart\_wkup} \)).

Refer to Table 172 for a detailed description of all USART interrupt requests.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Interrupt activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit data register empty</td>
<td>TXE</td>
<td>TXEIE</td>
<td>TXE cleared when a data is written in TDR</td>
<td>YES</td>
</tr>
<tr>
<td>Transmit FIFO Not Full</td>
<td>TXFNF</td>
<td>TXFNFIE</td>
<td>TXFNF cleared when TXFIFO is full.</td>
<td>YES</td>
</tr>
<tr>
<td>Transmit FIFO Empty</td>
<td>TXFE</td>
<td>TXFEIE</td>
<td>TXFE cleared when the TXFIFO contains at least one data or by setting TXFRQ bit.</td>
<td>YES</td>
</tr>
</tbody>
</table>
### Table 172. USART interrupt requests (continued)

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Interrupt activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFTIE</td>
<td>TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold</td>
<td>YES YES</td>
</tr>
<tr>
<td>CTS interrupt</td>
<td>CTSIF</td>
<td>CTSIE</td>
<td>CTSIF cleared by software by setting CTSCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Transmission Complete</td>
<td>TC</td>
<td>TCIE</td>
<td>TC cleared when a data is written in TDR or by setting TCCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Transmission Complete Before Guard Time</td>
<td>TCBGT</td>
<td>TCBGTIE</td>
<td>TCBGT cleared when a data is written in TDR or by setting TCBGTCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Receive data register not empty (data ready to be read)</td>
<td>RXNE</td>
<td>RXNEIE</td>
<td>RXNE cleared by reading RDR or by setting RXFRQ bit.</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO Not Empty</td>
<td>RXFNE</td>
<td>RXFNEIE</td>
<td>RXFNE cleared when the RXFIFO is empty or by setting RXFRQ bit.</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO Full</td>
<td>RXFF(^{(1)})</td>
<td>RXFFIE</td>
<td>RXFF cleared when the RXFIFO contains at least one data.</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFTIE</td>
<td>RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold</td>
<td>YES YES</td>
</tr>
<tr>
<td>Overrun error detected</td>
<td>ORE</td>
<td>RX-NEIE/RXFNEIE</td>
<td>ORE cleared by setting ORECFC bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Idle line detected</td>
<td>IDLE</td>
<td>IDLEIE</td>
<td>IDLE cleared by setting IDLEGFC bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Parity error</td>
<td>PE</td>
<td>PEIE</td>
<td>PE cleared by setting PECCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>LIN break</td>
<td>LBDF</td>
<td>LBDFIE</td>
<td>LBDF cleared by setting LBDFCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Noise Flag, Overrun error and Framing Error in multibuffer communication.</td>
<td>NE or ORE or FE</td>
<td>EIE</td>
<td>NE cleared by setting NECF bit. ORE cleared by setting ORECFC bit. FE flag cleared by setting FECFC bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Character match</td>
<td>CMF</td>
<td>CMIE</td>
<td>CMF cleared by setting CMCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Receiver timeout</td>
<td>RTOF</td>
<td>RTOFIE</td>
<td>RTOF cleared by setting RTOCCF bit.</td>
<td>YES NO</td>
</tr>
</tbody>
</table>
# Table 172. USART interrupt requests (continued)

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Interrupt activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>End of Block</td>
<td>EOBF</td>
<td>EOBIE</td>
<td>EOBF is cleared by setting EOBCF bit.</td>
<td>YES     NO</td>
</tr>
<tr>
<td>Wakeup from low-power mode</td>
<td>WUF</td>
<td>WUFIE</td>
<td>WUF is cleared by setting WUCF bit.</td>
<td>YES     YES</td>
</tr>
<tr>
<td>SPI slave underrun error</td>
<td>UDR</td>
<td>EIE</td>
<td>UDR is cleared by setting UDRCF bit.</td>
<td>YES     NO</td>
</tr>
<tr>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFTIE</td>
<td>TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold</td>
<td>YES     YES</td>
</tr>
<tr>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFTIE</td>
<td>RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold</td>
<td>YES     YES</td>
</tr>
</tbody>
</table>

1. RXFF flag is asserted if the USART receives \(n+1\) data (\(n\) being the RXFIFO size): \(n\) data in the RXFIFO and 1 data in USART\_RDR. In Stop mode, USART\_RDR is not clocked. As a result, this register will not be written and once \(n\) data are received and written in the RXFIFO, the RXFF interrupt will be asserted (RXFF flag is not set).
32.7 USART registers

Refer to Section 1.2 on page 50 for a list of abbreviations used in register descriptions.

32.7.1 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00
Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

**FIFO mode enabled**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RXFFIE: RXFIFO Full interrupt enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Interrupt inhibited</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: USART interrupt generated when RXFF=1 in the USART_ISR register</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>TXFEIE: TXFIFO empty interrupt enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Interrupt inhibited</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: USART interrupt generated when TXFE=1 in the USART_ISR register</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>FIFOEN: FIFO mode enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: FIFO mode is disabled.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: FIFO mode is enabled.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bitfield can only be written when the USART is disabled (UE=0).</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in Smartcard modes only. It must not be enabled in IrDA and LIN modes.</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>M1: Word length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M[1:0] = '00': 1 start bit, 8 Data bits, n Stop bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M[1:0] = '01': 1 start bit, 9 Data bits, n Stop bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M[1:0] = '10': 1 start bit, 7 Data bits, n Stop bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit can only be written when the USART is disabled (UE=0).</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: In 7-bits data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.</td>
<td></td>
</tr>
</tbody>
</table>
Bit 27  **EOBIE**: End of Block interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

*Note*: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 26  **RTOIE**: Receiver timeout interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

*Note*: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 32.4: USART implementation on page 953.

Bits 25:21  **DEAT[4:0]**: Driver Enable assertion time
This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE=0).

*Note*: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bits 20:16  **DEDT[4:0]**: Driver Enable deassertion time
This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE=0).

*Note*: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 15  **OVER8**: Oversampling mode
0: Oversampling by 16
1: Oversampling by 8

This bit can only be written when the USART is disabled (UE=0).

*Note*: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.

Bit 14  **CMIE**: Character match interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13  **MME**: Mute mode enable
This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.
0: Receiver in active mode permanently
1: Receiver can switch between Mute mode and active mode.

Bit 12  **M0**: Word length
This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE=0).
Bit 11  **WAKE**: Receiver wakeup method

This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line
1: Address mark

This bitfield can only be written when the USART is disabled (UE=0).

Bit 10  **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software.

Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled
1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE=0).

Bit 9  **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.

0: Even parity
1: Odd parity

This bitfield can only be written when the USART is disabled (UE=0).

Bit 8  **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited
1: USART interrupt generated whenever PE=1 in the USART_ISR register

Bit 7  **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited
1: USART interrupt generated whenever TXFNF =1 in the USART_ISR register

Bit 6  **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited
1: USART interrupt generated whenever TC=1 in the USART_ISR register

Bit 5  **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited
1: USART interrupt generated whenever ORE=1 or RXFNE=1 in the USART_ISR register

Bit 4  **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited
1: USART interrupt generated whenever IDLE=1 in the USART_ISR register
Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

**Note:** During transmission, a low pulse on the TE bit (‘0’ followed by ‘1’) sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to ‘1’. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

**Note:** It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.

If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

**Note:** To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In Smartcard mode, (SCEN = 1), the SCLK is always available when CLKEN = 1, regardless of the UE bit value.
32.7.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00
Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td><strong>FIFOEN</strong>: FIFO mode enable</td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: FIFO mode is disabled.</td>
</tr>
<tr>
<td></td>
<td>1: FIFO mode is enabled.</td>
</tr>
<tr>
<td></td>
<td>This bitfield can only be written when the USART is disabled (UE=0).</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in Smartcard modes only. It must not be enabled in IrDA and LIN modes.</td>
</tr>
<tr>
<td>28</td>
<td><strong>M1</strong>: Word length</td>
</tr>
<tr>
<td></td>
<td>This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.</td>
</tr>
<tr>
<td></td>
<td>M[1:0] = '00': 1 start bit, 8 Data bits, n Stop bit</td>
</tr>
<tr>
<td></td>
<td>M[1:0] = '01': 1 start bit, 9 Data bits, n Stop bit</td>
</tr>
<tr>
<td></td>
<td>M[1:0] = '10': 1 start bit, 7 Data bits, n Stop bit</td>
</tr>
<tr>
<td></td>
<td>This bit can only be written when the USART is disabled (UE=0).</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: In 7-bits data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.</td>
</tr>
<tr>
<td>27</td>
<td><strong>EOBIE</strong>: End of Block interrupt enable</td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt inhibited</td>
</tr>
<tr>
<td></td>
<td>1: USART interrupt generated when the EOBF flag is set in the USART_ISR register</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.</td>
</tr>
<tr>
<td>26</td>
<td><strong>RTOIE</strong>: Receiver timeout interrupt enable</td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt inhibited</td>
</tr>
<tr>
<td></td>
<td>1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.</td>
</tr>
</tbody>
</table>
|      | **Note**: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 32.4: USART implementation on page 953.
Bits 25:21 **DEAT[4:0]:** Driver Enable assertion time
This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).
This bitfield can only be written when the USART is disabled (UE=0).
*Note:* If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bits 20:16 **DEDT[4:0]:** Driver Enable deassertion time
This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).
If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.
This bitfield can only be written when the USART is disabled (UE=0).
*Note:* If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 15 **OVER8:** Oversampling mode
0: Oversampling by 16
1: Oversampling by 8
This bit can only be written when the USART is disabled (UE=0).
*Note:* In LIN, IrDA and Smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE:** Character match interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME:** Mute mode enable
This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.
0: Receiver in active mode permanently
1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0:** Word length
This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).
This bit can only be written when the USART is disabled (UE=0).

Bit 11 **WAKE:** Receiver wakeup method
This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.
0: Idle line
1: Address mark
This bitfield can only be written when the USART is disabled (UE=0).

Bit 10 **PCE:** Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software.
Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled
This bitfield can only be written when the USART is disabled (UE=0).
Bit 9  **PS:** Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.
0: Even parity  
1: Odd parity
This bitfield can only be written when the USART is disabled (UE=0).

Bit 8  **PEIE:** PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited  
1: USART interrupt generated whenever PE=1 in the USART_ISR register

Bit 7  **TXEIE:** Transmit data register empty
This bit is set and cleared by software.
0: Interrupt inhibited  
1: USART interrupt generated whenever TXE =1 in the USART_ISR register

Bit 6  **TCIE:** Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited  
1: USART interrupt generated whenever TC=1 in the USART_ISR register

Bit 5  **RXNEIE:** Receive data register not empty
This bit is set and cleared by software.
0: Interrupt inhibited  
1: USART interrupt generated whenever ORE=1 or RXNE=1 in the USART_ISR register

Bit 4  **IDLEIE:** IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited  
1: USART interrupt generated whenever IDLE=1 in the USART_ISR register

Bit 3  **TE:** Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled  
1: Transmitter is enabled

**Note:** During transmission, a low pulse on the TE bit (‘0’ followed by ‘1’) sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to ‘1’. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.
Bit 2 **RE**: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode
When this bit is cleared, the USART cannot wake up the MCU from low-power mode.
When this bit is set, the USART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: USART not able to wake up the MCU from low-power mode.
1: USART able to wake up the MCU from low-power mode.

**Note**: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.
If the USART does not support the wake up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 0 **UE**: USART enable
When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.
0: USART prescaler and outputs disabled, low-power mode
1: USART enabled

**Note**: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.
In Smartcard mode, (SCEN = 1), the SCLK is always available when CLKEN = 1, regardless of the UE bit value.

### 32.7.3 USART control register 2 (USART_CR2)

Address offset: 0x04
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add[7:0]</td>
<td>RTOEN</td>
<td>ABRMOD[1:0]</td>
<td>ABREN</td>
<td>MSBF</td>
<td>RST</td>
<td>DATAINV</td>
<td>TXINV</td>
<td>RXINV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWAP</td>
<td>LINEN</td>
<td>STOP[1:0]</td>
<td>CLKEN</td>
<td>CPOL</td>
<td>CPHA</td>
<td>LBCL</td>
<td>Res.</td>
<td>LBDE</td>
<td>LBDL</td>
<td>ADDMT</td>
<td>DIS_</td>
<td>NSS_</td>
<td>Res.</td>
<td>Res.</td>
<td>SLVEN</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:24  **ADD[7:0]:** Address of the USART node

**ADD[7:4]:**
These bits give the address of the USART node or a character code to be recognized. They are used to wake up the MCU with 7-bit address mark detection in multiprocessor communication during Mute mode or low-power mode. The MSB of the character sent by the transmitter should be equal to 1. They can also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.

These bits can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0).

**ADD[3:0]:**
These bits give the address of the USART node or a character code to be recognized. They are used for wakeup with address mark detection, in multiprocessor communication during Mute mode or low-power mode.

These bits can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0).

Bit 23  **RTOEN:** Receiver timeout enable
This bit is set and cleared by software.

0: Receiver timeout feature disabled.
1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

**Note:** If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bits 22:21  **ABRMOD[1:0]:** Auto baud rate mode
These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.
01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)
10: 0x7F frame detection.
11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE=0).

**Note:** If DATAINV=1 and/or MSBFIRST=1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 20  **ABREN:** Auto baud rate enable
This bit is set and cleared by software.

0: Auto baud rate detection is disabled.
1: Auto baud rate detection is enabled.

**Note:** If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 19  **MSBFIRST:** Most significant bit first
This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE=0).
Bit 18  **DATAINV**: Binary data inversion
This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The
parity bit is also inverted.
This bitfield can only be written when the USART is disabled (UE=0).

Bit 17  **TXINV**: TX pin active level inversion
This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)
1: TX pin signal values are inverted. (V_{DD} =0/mark, Gnd=1/idle).
This allows the use of an external inverter on the TX line.
This bitfield can only be written when the USART is disabled (UE=0).

Bit 16  **RXINV**: RX pin active level inversion
This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)
1: RX pin signal values are inverted. (V_{DD} =0/mark, Gnd=1/idle).
This allows the use of an external inverter on the RX line.
This bitfield can only be written when the USART is disabled (UE=0).

Bit 15  **SWAP**: Swap TX/RX pins
This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This allows to work in the case of a cross-wired
connection to another UART.
This bitfield can only be written when the USART is disabled (UE=0).

Bit 14  **LINEN**: LIN mode enable
This bit is set and cleared by software.
0: LIN mode disabled
1: LIN mode enabled
The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the
SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.
This bitfield can only be written when the USART is disabled (UE=0).

*Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value.*
*Refer to Section 32.4: USART implementation on page 953.*

Bits 13:12  **STOP[1:0]**: stop bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: 0.5 stop bit.
10: 2 stop bits
11: 1.5 stop bits
This bitfield can only be written when the USART is disabled (UE=0).
Bit 11 **CLKEN**: Clock enable

This bit allows the user to enable the SCLK pin.

0: SCLK pin disabled
1: SCLK pin enabled

This bit can only be written when the USART is disabled (UE=0).

Note: If neither synchronous mode nor Smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

In Smartcard mode, in order to provide correctly the SCLK clock to the smartcard, the steps below must be respected:

- UE = 0
- SCEN = 1
- GTPR configuration
- CLKEN= 1
- UE = 1

Bit 10 **CPOL**: Clock polarity

This bit allows the user to select the polarity of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship.

0: Steady low value on SCLK pin outside transmission window
1: Steady high value on SCLK pin outside transmission window

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see Figure 321 and Figure 322).

0: The first clock transition is the first data capture edge
1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the SCLK pin
1: The clock pulse of the last data bit is output to the SCLK pin

**Caution:** The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited
1: An interrupt is generated whenever LBDF=1 in the USART_ISR register

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.
Bit 5 **LBDL**: LIN break detection length
This bit is for selection between 11 bit or 10 bit break detection.
0: 10-bit break detection
1: 11-bit break detection
This bit can only be written when the USART is disabled (UE=0).
*Note:* If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 4 **ADDM7**: 7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the USART is disabled (UE=0)
*Note:* In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**: When the DIS_NSS bit is set, the NSS pin input is ignored.
0: SPI slave selection depends on NSS input pin.
1: SPI slave is always selected and NSS input pin is ignored.
*Note:* When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous Slave mode enable
When the SLVEN bit is set, the synchronous slave mode is enabled.
0: Slave mode disabled.
1: Slave mode enabled.
*Note:* When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

*Note:* The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.

### 32.7.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DEP</td>
<td>DEM</td>
<td>DDRE</td>
<td>OVR</td>
<td>ONE BIT</td>
<td>CTSIE</td>
<td>CTSE</td>
<td>RTSE</td>
<td>DMAT</td>
<td>DMAR</td>
<td>SCEN</td>
<td>NACK</td>
<td>HD SEL</td>
<td>IRLP</td>
<td>IREN</td>
<td>EIE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
- 000: TXFIFO reaches 1/8 of its depth
- 001: TXFIFO reaches 1/4 of its depth
- 010: TXFIFO reaches 1/2 of its depth
- 011: TXFIFO reaches 3/4 of its depth
- 100: TXFIFO reaches 7/8 of its depth
- 101: TXFIFO becomes empty
- Remaining combinations: Reserved

Bit 28 **RXFTIE:** RXFIFO threshold interrupt enable
This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
- 000: Receive FIFO reaches 1/8 of its depth
- 001: Receive FIFO reaches 1/4 of its depth
- 010: Receive FIFO reaches 1/2 of its depth
- 011: Receive FIFO reaches 3/4 of its depth
- 100: Receive FIFO reaches 7/8 of its depth
- 101: Receive FIFO becomes full
- Remaining combinations: Reserved

Bit 24 **TCBGTIE:** Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated whenever TCBGT=1 in the USART_ISR register

*Note:* If the USART does not support the Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 23 **TXFTIE:** TXFIFO threshold interrupt enable
This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE:** Wakeup from low-power mode interrupt enable
This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated whenever WUF=1 in the USART_ISR register

*Note:* WUFIE must be set before entering in low-power mode.

*If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.*
Bits 21:20 **WUS[1:0]:** Wakeup from low-power mode interrupt flag selection

- This bitfield specifies the event which activates the WUF (Wakeup from low-power mode flag).
- 00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
- 01: Reserved.
- 10: WUF active on start bit detection
- 11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the USART is disabled (UE=0).

*If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.*

Bits 19:17 **SCARCNT[2:0]:** Smartcard auto.retry count

- This bitfield specifies the number of retries for transmission and reception in Smartcard mode.
- In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).
- In reception mode, it specifies the number of erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE=0).

When the USART is enabled (UE=1), this bitfield may only be written to 0x0, in order to stop retransmission.

- 0x0: retransmission disabled - No automatic retransmission in transmit mode.
- 0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

*Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.*

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP: Driver enable polarity selection**

- 0: DE signal is active high.
- 1: DE signal is active low.

This bit can only be written when the USART is disabled (UE=0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.*

Bit 14 **DEM: Driver enable mode**

- This bit allows the user to activate the external transceiver control, through the DE signal.
- 0: DE function is disabled.
- 1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE=0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 32.4: USART implementation on page 953.*

Bit 13 **DDRE: DMA Disable on Reception Error**

- 0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data will be transferred. (used for Smartcard mode)
- 1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE/RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE=0).

*Note: The reception errors are: parity error, framing error or noise error.*
Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set, the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO will be bypassed and data will be written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.
This bit can only be written when the USART is disabled (UE=0).

*Note: This control bit allows checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable
This bit allows the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.
0: Three sample bit method
1: One sample bit method
This bit can only be written when the USART is disabled (UE=0).

Bit 10 **CTSE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF=1 in the USART_ISR register

*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.*

Bit 9 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.
This bit can only be written when the USART is disabled (UE=0).

*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.*

Bit 8 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 7 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception
Bit 5 **SCEN**: Smartcard mode enable
This bit is used for enabling Smartcard mode.
0: Smartcard Mode disabled
1: Smartcard Mode enabled
This bitfield can only be written when the USART is disabled (UE=0).
*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.*

Bit 4 **NACK**: Smartcard NACK enable
0: NACK transmission in case of parity error is disabled
1: NACK transmission during parity error is enabled
This bitfield can only be written when the USART is disabled (UE=0).
*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.*

Bit 3 **HDSEL**: Half-duplex selection
Selection of Single-wire Half-duplex mode
0: Half duplex mode is not selected
1: Half duplex mode is selected
This bit can only be written when the USART is disabled (UE=0).

Bit 2 **IRLP**: IrDA low-power
This bit is used for selecting between normal and low-power IrDA modes
0: Normal mode
1: Low-power mode
This bit can only be written when the USART is disabled (UE=0).
*Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.*

Bit 1 **IREN**: IrDA mode enable
This bit is set and cleared by software.
0: IrDA disabled
1: IrDA enabled
This bit can only be written when the USART is disabled (UE=0).
*Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.*

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE=1 or ORE=1 or NE=1or UDR = 1 in the USART_ISR register).
0: Interrupt inhibited
1: interrupt generated when FE=1 or ORE=1 or NE=1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.
32.7.5 USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C
Reset value: 0x0000 0000

| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0  | **BRR[15:0]**: USART baud rate |

**BRR[15:4]**


**BRR[3:0]**

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].
When OVER8 = 1:
BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
BRR[3] must be kept cleared.

32.7.6 USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10
Reset value: 0x0000 0000

| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0  | **GT[7:0]**, **PSC[7:0]** |

**GT[7:0]**, **PSC[7:0]**
32.7.7 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 GT[7:0]: Guard time value
This bitfield is used to program the Guard time value in terms of number of baud clock periods.
This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.
This bitfield can only be written when the USART is disabled (UE=0).
Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value.
Refer to Section 32.4: USART implementation on page 953.

Bits 7:0 PSC[7:0]: Prescaler value

In IrDA low-power and normal IrDA mode:
PSC[7:0] = IrDA Normal and Low-Power Baud Rate
Used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:
The source clock is divided by the value given in the register (8 significant bits):
00000000: Reserved - do not program this value
00000001: divides the source clock by 1
00000010: divides the source clock by 2
...

In Smartcard mode:
PSC[4:0]: Prescaler value
Used for programming the prescaler for dividing the USART source clock to provide the Smartcard clock.
The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:
00000: Reserved - do not program this value
00001: divides the source clock by 2
00010: divides the source clock by 4
00011: divides the source clock by 6
...
This bitfield can only be written when the USART is disabled (UE=0).
Note: Bits [7:5] must be kept cleared if Smartcard mode is used.
This bitfield is reserved and forced by hardware to '0' when the Smartcard and IrDA modes are not supported. Refer to Section 32.4: USART implementation on page 953.
Bits 31:24  **BLEN[7:0]: Block Length**

This bitfield gives the Block length in Smartcard T=1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:
- \(BLEN = 0\) -> 0 information characters + LEC
- \(BLEN = 1\) -> 0 information characters + CRC
- \(BLEN = 255\) -> 254 information characters + CRC (total 256 characters)

In Smartcard mode, the Block length counter is reset when TXE=0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the Block length counter is reset when RE=0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note:  *This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.*

Bits 23:0  **RTO[23:0]: Receiver timeout value**

This bitfield gives the Receiver timeout value in terms of number of bits during which there is no activity on the RX line.

In standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In Smartcard mode, this value is used to implement the CWT and BWT. See Smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

Note:  *This value must only be programmed once per received character.*

**Note:**  RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to “0x00000000” when the Receiver timeout feature is not supported. Refer to Section 32.4: USART implementation on page 953.

### 32.7.8 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>
32.7.9 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/Smartcard mode is enabled

XX = 08 if FIFO is enabled and Smartcard mode is disabled

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

### FIFO mode enabled

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

### Bits 31:5
Reserved, must be kept at reset value.

### Bit 4 TXFRQ: Transmit data flush request

When FIFO mode is disabled, writing ‘1’ to this bit sets the TXE flag. This allows to discard the transmit data. This bit must be used only in Smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and Smartcard modes.

**Note:** In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

### Bit 3 RXFRQ: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO i.e. clears the bit RXFNE. This allows to discard the received data without reading them, and avoid an overrun condition.

### Bit 2 MMRQ: Mute mode request

Writing 1 to this bit puts the USART in Mute mode and resets the RWU flag.

### Bit 1 SBKRQ: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

**Note:** When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.

### Bit 0 ABRQQ: Auto baud rate request

Writing 1 to this bit resets the ABRF flag in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

**Note:** If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.
Bits 31:28  Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag
This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG of USART_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit =1 (bit 31) in the USART_CR3 register.
0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag
This bit is set by hardware when the threshold programmed in RXFTCFG in USART_CR3 register is reached. This means that there are (RXFTCFG - 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit =1 (bit 27) in the USART_CR3 register.
0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Note: **When the RXFTCFG threshold is configured to ‘101’, RXFT flag will be set if 16 data are available i.e. 15 data in the RXFIFO and 1 data in the USART_RDR. Consequently, the 17th received data will not cause an overrun error. The overrun error occurs after receiving the 18th data.**

Bit 25 **TCBGT**: Transmission complete before guard time flag
This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.
It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE=1 in the USART_CR3 register.
This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)
1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

Note: **If the USART does not support the Smartcard mode, this bit is reserved and kept at reset value. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is ‘1’. Refer to Section 32.4: USART implementation on page 953.**

Bit 24 **RXFF**: RXFIFO full
This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.
An interrupt is generated if the RXFFIE bit =1 in the USART_CR1 register.
0: RXFIFO not full.
1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO empty
This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.
An interrupt is generated if the TXFEIE bit =1 (bit 30) in the USART_CR1 register.
0: TXFIFO not empty.
1: TXFIFO empty.
Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART. It can be used to verify that the USART is ready for reception before entering low-power mode.

*Note*: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART. It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register. An interrupt is generated if WUFIE=1 in the USART_CR3 register.

*Note*: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 19 **RWU**: Receiver wakeup from Mute mode

This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register. When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode
1: Receiver in Mute mode

*Note*: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character is transmitted
1: Break character will be transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register. An interrupt is generated if CMIE=1 in the USART_CR1 register.

0: No Character match detected
1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)
1: Reception on going
Bit 15 **ABRF**: Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE will also be set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXFNE and FE are also set in this case). It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

**Note**: *If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed) It is cleared by software, by writing 1 to the ABRRQ bit in the USART_CR3 register.

**Note**: *If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error
1: underrun error

**Note**: *If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 32.4: USART implementation on page 953.*

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T=1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4. An interrupt is generated if the EOBIE=1 in the USART_CR2 register. It is cleared by software, writing 1 to the EOBCF in the USART_ICR register.

0: End of Block not reached
1: End of Block (number of characters) reached

**Note**: *If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 32.4: USART implementation on page 953.*

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE=1 in the USART_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached
1: Timeout value reached without any data reception

**Note**: *If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set. The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF will be set. If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.*
Bit 10 **CTS**: CTS flag
This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.
0: nCTS line set
1: nCTS line reset

*Note:* If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag
This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.
An interrupt is generated if CTSIE=1 in the USART_CR3 register.
0: No change occurred on the nCTS status line
1: A change occurred on the nCTS status line

*Note:* If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag
This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.
An interrupt is generated if LBDIE = 1 in the USART_CR2 register.
0: LIN Break not detected
1: LIN break detected

*Note:* If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 7 **TXFNF**: TXFIFO not full
TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR. An interrupt is generated if the TXFNFIE bit =1 in the USART_CR1 register.
0: Transmit FIFO is full
1: Transmit FIFO is not full

*Note:* The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE will be set at the same time).

This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete
This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.
It is set by hardware when the transmission of a frame containing data is complete and when TXFE is set.
An interrupt is generated if TCIE=1 in the USART_CR1 register.
TC bit is cleared by software, by writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission is not complete
1: Transmission is complete

*Note:* If TE bit is reset and no transmission is on going, the TC bit will be set immediately.
Bit 5 **RXFNE**: RXFIFO not empty
RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.
RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.
An interrupt is generated if RXFNEIE=1 in the USART_CR1 register.
0: Data is not received
1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected
This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.
0: No Idle line is detected
1: Idle line is detected

Note: The IDLE bit will not be set again until the RXFNE bit has been set (i.e. a new idle line occurs).
If Mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error
This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by software, writing 1 to the ORECF, in the USART_ICR register.
An interrupt is generated if RXFNEIE=1 or EIE = 1 in the USART_CR1 register.
0: No overrun error
1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.
This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.
Bit 2 **NE**: Noise detection flag
This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the USART_ICR register.
0: No noise is detected
1: Noise is detected

*Note:* This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 32.5.8: Tolerance of the USART receiver to clock deviation on page 971).
This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error
This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.
When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the USART_CR1 register.
0: No Framing error is detected
1: Framing error or break character is detected

*Note:* This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error
This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.
An interrupt is generated if PEIE = 1 in the USART_CR1 register.
0: No parity error
1: Parity error

*Note:* This error is associated with the character in the USART_RDR.

### 32.7.10 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C
Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

#### FIFO mode disabled

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>----</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ABRF</td>
<td>ABRE</td>
<td>UDR</td>
<td>EOBF</td>
<td>RTOF</td>
<td>CTS</td>
<td>CTSIF</td>
<td>LBDF</td>
<td>TXE</td>
<td>TC</td>
<td>RXNE</td>
<td>IDLE</td>
<td>ORE</td>
<td>NE</td>
<td>FE</td>
<td>PE</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT**: Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE=1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)
1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

**Note:** If the USART does not support the Smartcard mode, this bit is reserved and kept at reset value. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is ‘1’. Refer to Section 32.4: USART implementation on page 953.

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

**Note:** If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE=1 in the USART_CR3 register.

**Note:** When UESM is cleared, WUF flag is also cleared.

**If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 32.4: USART implementation on page 953.**

Bit 19 **RWU**: Receiver wakeup from Mute mode

This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode
1: Receiver in Mute mode

**Note:** If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 32.4: USART implementation on page 953.
Bit 18 **SBKF**: Send break flag
This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.
0: No break character is transmitted
1: Break character will be transmitted

Bit 17 **CMF**: Character match flag
This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register. An interrupt is generated if CMIE=1 in the USART_CR1 register.
0: No Character match detected
1: Character Match detected

Bit 16 **BUSY**: Busy flag
This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).
0: USART is idle (no reception)
1: Reception ongoing

Bit 15 **ABRF**: Auto baud rate flag
This bit is set by hardware when the automatic baud rate has been set (RXNE will also be set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXNE and FE are also set in this case). It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRQR in the USART_RQR register.

Note: *If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 14 **ABRE**: Auto baud rate error
This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed). It is cleared by software, by writing 1 to the ABRQR bit in the USART_CR3 register.

Note: *If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 13 **UDR**: SPI slave underrun error flag
In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.
0: No underrun error
1: underrun error

Note: *If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 32.4: USART implementation on page 953.*

Bit 12 **EOBF**: End of block flag
This bit is set by hardware when a complete block has been received (for example T=1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.
An interrupt is generated if the EOBIE=1 in the USART_CR2 register.
It is cleared by software, writing 1 to the EOBCF in the USART_ICR register.
0: End of Block not reached
1: End of Block (number of characters) reached

Note: *If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 32.4: USART implementation on page 953.*
Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE=1 in the USART_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached
1: Timeout value reached without any data reception

*Note:* If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF will be set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set
1: nCTS line reset

*Note:* If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE=1 in the USART_CR3 register.

0: No change occurred on the nCTS status line
1: A change occurred on the nCTS status line

*Note:* If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected
1: LIN break detected

*Note:* If the USART does not support LIN mode, this bit is reserved and kept at reset value.

Refer to Section 32.4: USART implementation on page 953.

Bit 7 **TXE**: Transmit data register empty

TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in Smartcard T=0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register.

0: Data register full
1: Data register not full
Bit 6 **TC:** Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

- It is set by hardware when the transmission of a frame containing data is complete and when TXE is set.
- An interrupt is generated if TCIE=1 in the USART_CR1 register.
- TC bit is is cleared by software, by writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete
1: Transmission is complete

**Note:** If TE bit is reset and no transmission is on going, the TC bit will be set immediately.

Bit 5 **RXNE:** Read data register not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

- An interrupt is generated if RXNEIE=1 in the USART_CR1 register.

0: Data is not received
1: Received data is ready to be read

Bit 4 **IDLE:** Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected
1: Idle line is detected

**Note:** The IDLE bit will not be set again until the RXNE bit has been set (i.e. a new idle line occurs).

If Mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE:** Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE=1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

- An interrupt is generated if RXNEIE=1 or EIE = 1 in the USART_CR1 register.

0: No overrun error
1: Overrun error is detected

**Note:** When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.
Bit 2 **NE**: Noise detection flag
This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the USART_ICR register.

0: No noise is detected
1: Noise is detected

*Note:* This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 32.5.8: Tolerance of the USART receiver to clock deviation on page 971).

Bit 1 **FE**: Framing error
This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR1 register.

0: No Framing error is detected
1: Framing error or break character is detected

Bit 0 **PE**: Parity error
This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error
1: Parity error

### 32.7.11 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20
Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>UDRCF</td>
<td>EOBCF</td>
<td>RTOCF</td>
<td>Res</td>
<td>Res</td>
<td>CTSCF</td>
<td>LBCF</td>
<td>TCBGT</td>
<td>CF</td>
<td>TCCF</td>
<td>TXXFEC</td>
<td>F</td>
<td>IDLECF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wakeup from low-power mode clear flag
Writing 1 to this bit clears the WUF flag in the USART_ISR register.

*Note:* If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag
Writing 1 to this bit clears the CMF flag in the USART_ISR register.
Bits 16:14  Reserved, must be kept at reset value.

Bit 13 **UDRCF**: SPI slave underrun clear flag
Writing 1 to this bit clears the UDRF flag in the USART_ISR register.

*Note*: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 12 **EOBCF**: End of block clear flag
Writing 1 to this bit clears the EOBF flag in the USART_ISR register.

*Note*: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 11 **RTOCF**: Receiver timeout clear flag
Writing 1 to this bit clears the RTOF flag in the USART_ISR register.

*Note*: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.

*Note*: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.

*Note*: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation on page 953.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.
32.7.12 USART receive data register (USART_RDR)

Address offset: 0x24
Reset value: 0xXXXX XXXX

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see Figure 315).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

32.7.13 USART transmit data register (USART_TDR)

Address offset: 0x28
Reset value: 0xXXXX XXXX

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value
Contains the data character to be transmitted.
The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 315).
When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

*Note: This register must be written only when TXE/TXENF=1.*
32.7.14 USART prescaler register (USART_PRES)

This register can only be written when the USART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000 0000

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


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

Bits 3:0 PRESCALER[3:0]: Clock prescaler

The USART input clock can be divided by a prescaler factor:

- 0000: input clock not divided
- 0001: input clock divided by 2
- 0010: input clock divided by 4
- 0011: input clock divided by 6
- 0100: input clock divided by 8
- 0101: input clock divided by 10
- 0110: input clock divided by 12
- 0111: input clock divided by 16
- 1000: input clock divided by 32
- 1001: input clock divided by 64
- 1010: input clock divided by 128
- 1011: input clock divided by 256

Remaining combinations: Reserved

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value will be ‘1011’ i.e. input clock divided by 256.
32.7.15 USART register map

The table below gives the USART register map and reset values.

Table 173. USART register map and reset values

| Offset | Register name          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |  |
|---------|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00    | USART_CR1             | RXFFIE | TXFFIE | FIFOEN | M1 | EOIE | RTOIE | DEAT[4:0] | DEDT[4:0] | OVER8 | CMIE | MME | M0 | WAKE | PCE | PS | PE | PE | TXFNFIE | TCIE | RXFNEIE | RXNEIE | IDLEIE | TE | RE | UESM | UE |  |
|         | FIFO enabled          | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| 0x00    | USART_CR1             | RXFFIE | TXFFIE | FIFOEN | M1 | EOIE | RTOIE | DEAT[4:0] | DEDT[4:0] | OVER8 | CMIE | MME | M0 | WAKE | PCE | PS | PE | PE | TXFNFIE | TCIE | RXFNEIE | RXNEIE | IDLEIE | TE | RE | UESM | UE |  |
|         | FIFO disabled         | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
|         |                       | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| 0x08    | USART_CR3             | TXFTG[2:0] | RXFTIE | TXFTG[2:0] | RXFTIE | TCBGIE | WUIE | RTOEN | ABLO [1:0] | ARBREN | HSERST | DATAW | TXINV | RXINV | STRO | OVRDIS | ONEBIT | CTIE | CTSE | CTSE | RTSE | RTSE | RTSE | RTSE | RTSE | RTSE | RTSE | RTSE | RTSE | RTSE | RTSE | RTSE |  |
|         |                       | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
|         |                       | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
|         |                       | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
|         |                       | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
|         |                       | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
Table 173. USART register map and reset values (continued)

| Offset | Register name      | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|        | FIFO mode         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | enabled           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 0 X 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 |
|        | FIFO mode         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | disabled          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 |

Refer to Section 2.2: Memory organization for the register boundary addresses.
33 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitted (LPUART).

33.1 LPUART introduction

The LPUART is an UART which allows bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to allow UART communications up to 9600 baud/s. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports Half-duplex Single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.
33.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 baud/s to 9600 baud/s using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data
  Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wakeup from Mute mode by idle line detection or address mark detection

33.3 LPUART implementation

The following table shows LPUART implementation in comparison with USART.
# Table 174. LPUART features

<table>
<thead>
<tr>
<th>LPUART modes/features(^{(1)})</th>
<th>LPUART</th>
<th>USART1/2</th>
<th>USART3/4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware flow control for modem</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Continuous communication using DMA</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Multiprocessor communication</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Synchronous mode (Master/Slave)</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Smartcard mode</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Single-wire Half-duplex communication</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IrDA SIR ENDEC block</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>LIN mode</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Dual clock domain and wakeup from low-power mode</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Receiver timeout interrupt</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Modbus communication</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Auto baud rate detection</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Driver Enable</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>USART data length</strong></td>
<td><strong>7, 8 and 9 bits</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tx/Rx FIFO</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Tx/Rx FIFO size</td>
<td>8</td>
<td>8</td>
<td>-</td>
</tr>
</tbody>
</table>

1. X = supported.
33.4 LPUART functional description

33.4.1 LPUART block diagram

The simplified block diagram given in Figure 342 shows two fully independent clock domains:

- The **lpuart_pclk** clock domain
  - The **lpuart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.

- The **lpuart_ker_ck** kernel clock domain
  - The **lpuart_ker_ck** is the LPUART clock source. It is independent of the **lpuart_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart_ker_ck** is stopped.
  
  When the dual clock domain feature is disabled, the **lpuart_ker_ck** is the same as the **lpuart_pclk** clock.

There is no constraint between **lpuart_pclk** and **lpuart_ker_ck**: **lpuart_ker_ck** can be faster or slower than **lpuart_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.
33.4.2 LPUART signals

LPUART bidirectional communications requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)
  
  RX is the serial data input.

- **TX** (Transmit Data Output)
  
  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In Single-wire mode, this I/O is used to transmit and receive the data.

RS232 hardware flow control mode

The following pins are required in RS232 Hardware flow control mode:

- **CTS** (Clear To Send)
  
  When driven high, this signal blocks the data transmission at the end of the current transfer.

- **RTS** (Request to send)
  
  When it is low, this signal indicates that the USART is ready to receive data.

RS485 hardware flow control mode

The following pin is required in RS485 Hardware control mode:

- **DE** (Driver Enable)
  
  This signal activates the transmission mode of the external transceiver.

Note: **DE and RTS share the same pin.**

33.4.3 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see Figure 316).

- 7-bit character length: \( M[1:0] = '10' \)
- 8-bit character length: \( M[1:0] = '00' \)
- 9-bit character length: \( M[1:0] = '01' \)

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

An **Idle character** is interpreted as an entire frame of “1”s. (The number of “1” ‘s will include the number of stop bits).

A **Break character** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.
33.4.4 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bits wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store...
the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

**Note:** The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR. The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in LPUART_CR3 control register.

In this case:

- The RXFT flag is set in the LPUART_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bit fields. This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.
  RXFTCFG data have been received: one data in LPUART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to ‘101’, the RXFT flag will be set when a number of data corresponding to the FIFO size has been received: FIFO size - 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data will not set the overrun flag.

- The TXFT flag is set in the LPUART_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bit fields.
  This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

### 33.4.5 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

**Character transmission**

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see Figure 342).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

**Note:** The TE bit must be set before writing the data to be transmitted to the LPUART_TDR. The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission will corrupt the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost. An idle frame will be sent after the TE bit is enabled.
Configurable stop bits

The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13,12).

- **1 stop bit**: This is the default value of number of stop bits.
- **2 Stop bits**: This will be supported by normal LPUART, Single-wire and Modem modes.

An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits (when M[1:0] = ‘00’) or 11 low bits (when M[1:0] = ‘01’) or 9 low bits (when M[1:0] = ‘10’) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 344. Configurable stop bits

Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to ‘1’.
5. Select DMA enable (DMAT) in LPUART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in **Section 32.5.10: USART multiprocessor communication**.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
   - When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   - When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART_TDR register, wait until TC=1. This indicates that the transmission of the last frame is complete.
   - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

**Single byte communication**

- **When FIFO mode disabled:**
  
  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:
  
  - the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;
  - the LPUART_TDR register is empty;
  - the next data can be written to the LPUART_TDR register without overwriting the previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- **When FIFO mode is enabled,** the TXFNF (TXFIFO Not Full) flag is set by hardware to indicate that:
  
  - the TXFIFO is not full;
  - the LPUART_TDR register is empty;
  - the next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if TXFNEIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE is case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

  After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC=1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see *Figure 345: TC/TXE behavior when transmitting*).
Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

**Break characters**

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see Figure 343).

If a ‘1’ is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

**Idle characters**

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

### 33.4.6 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

**Start bit detection**

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still ‘0’. If the start sample is at ‘1’, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.
Character reception
During an LPUART reception, data are shifted in least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure
To receive a character, follow the sequence below:
1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to ‘1’.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in Section 32.5.10: USART multiprocessor communication.
6. Set the RE bit LPUART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received
- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the Receive Data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty i.e. there is a data in the RXFIFO to be read.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set.
Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

**Break character**

When a break character is received, the USART handles it as a framing error.

**Idle character**

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

**Overrun error**

- **FIFO mode disabled**
  An overrun error occurs when a character is received when RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.
  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
  - the ORE bit is set;
  - the RDR content will not be lost. The previous data is available when a read to LPUART_RDR is performed.;
  - the shift register will be overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXNEIE bit or EIE bit is set.

- **FIFO mode enabled**
  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.
  Data can not be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.
  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:
  - the ORE bit is set;
  - the first entry in the RXFIFO will not be lost. It is available when a read to LPUART_RDR is performed.
  - the shift register will be overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.
Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities
- if RXNE=1, then the last valid data is stored in the receive register (RDR) and can be read,
- if RXNE=0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.

Selecting the clock source

The choice of the clock source is done through the Clock Control system (see Section Reset and clock controller (RCC)). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wakeup from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see Section Reset and clock controller (RCC)). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRES register.

Some lpuart_ker_ck sources allow the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to allow LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming baud as close as possible to the middle of the baud-period. Only a single sample is taken of each of the incoming bauds.

Note: There is no noise detection for data.
Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:
- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the LPUART_RDR register.
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication, an interrupt will be issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- 1 stop bit: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- 2 stop bits: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample i.e. during the second stop bit. The first stop bit is not checked for framing error.

33.4.7 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register:

\[
\text{Tx/Rx baud} = \frac{256 \times \text{lpuartckpres}}{\text{LPUARTDIV}}
\]

LPUARTDIV is defined in the LPUART_BRR register.

Note:

* The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence the baud rate register value should not be changed during communication.
* It is forbidden to write values lower than 0x300 in the LPUART_BRR register.
* \(f_{CK}\) must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 baud. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock. For example, if the LPUART clock source frequency is 100 MHz, the maximum baud rate that can be reached is about 33 Mbaud.
### Table 175. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32,768 KHz

<table>
<thead>
<tr>
<th>S.No</th>
<th>Desired</th>
<th>Actual</th>
<th>Value programmed in the baud rate register</th>
<th>% Error = (Calculated - Desired) B.rate / Desired B.rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3 KBps</td>
<td>0.3 KBps</td>
<td>0x6D3A</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.6 KBps</td>
<td>0.6 KBps</td>
<td>0x369D</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1200 Bps</td>
<td>1200.087 Bps</td>
<td>0x1B4E</td>
<td>0.007</td>
</tr>
<tr>
<td>4</td>
<td>2400 Bps</td>
<td>2400.17 Bps</td>
<td>0xDA7</td>
<td>0.007</td>
</tr>
<tr>
<td>5</td>
<td>4800 Bps</td>
<td>4801.72 Bps</td>
<td>0x6D3</td>
<td>0.035</td>
</tr>
<tr>
<td>6</td>
<td>9600 KBps</td>
<td>9608.94 Bps</td>
<td>0x369</td>
<td>0.093</td>
</tr>
</tbody>
</table>

### Table 176. Error calculation for programmed baud rates at fCK = 100 MHz

<table>
<thead>
<tr>
<th>S.No</th>
<th>Desired</th>
<th>Actual</th>
<th>Value programmed in the baud rate register</th>
<th>% Error = (Calculated - Desired) B(rate) / Desired B(rate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>38400 Baud</td>
<td>38400,04 Baud</td>
<td>A2C2A</td>
<td>0.0001</td>
</tr>
<tr>
<td>2</td>
<td>57600 Baud</td>
<td>57600,06 Baud</td>
<td>6C81C</td>
<td>0.0001</td>
</tr>
<tr>
<td>3</td>
<td>115200 Baud</td>
<td>115200,12 Baud</td>
<td>3640E</td>
<td>0.0001</td>
</tr>
<tr>
<td>4</td>
<td>230400 Baud</td>
<td>230400,23 Baud</td>
<td>1B207</td>
<td>0.0001</td>
</tr>
<tr>
<td>5</td>
<td>460800 Baud</td>
<td>460804,61 Baud</td>
<td>D903</td>
<td>0.001</td>
</tr>
<tr>
<td>6</td>
<td>921600 Baud</td>
<td>921625,81 Baud</td>
<td>6C81</td>
<td>0.0028</td>
</tr>
<tr>
<td>7</td>
<td>4000 KBaud</td>
<td>4000000.00 Baud</td>
<td>1900</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>10000 Kbaud</td>
<td>10000000.00 Baud</td>
<td>A00</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>20000 Kbaud</td>
<td>20000000.00 Baud</td>
<td>500</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>30000 Kbaud</td>
<td>33032258,06 Baud</td>
<td>307</td>
<td>0.1</td>
</tr>
</tbody>
</table>
33.4.8 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter’s local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

\[
\text{DTRA + DQUANT + DREC + DTCL + DWU < LPUART receiver tolerance}
\]

where

\[
\text{DWU is the error due to sampling point deviation when the wakeup from low-power mode is used.}
\]

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in Table 177:

- Number of Stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

<table>
<thead>
<tr>
<th>M bits</th>
<th>768 &lt; BRR &lt; 1024</th>
<th>1024 &lt; BRR &lt; 2048</th>
<th>2048 &lt; BRR &lt; 4096</th>
<th>4096 ≤ BRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits (M=’00’), 1 Stop bit</td>
<td>1.82%</td>
<td>2.56%</td>
<td>3.90%</td>
<td>4.42%</td>
</tr>
<tr>
<td>9 bits (M=’01’), 1 Stop bit</td>
<td>1.69%</td>
<td>2.33%</td>
<td>2.53%</td>
<td>4.14%</td>
</tr>
<tr>
<td>7 bits (M=’10’), 1 Stop bit</td>
<td>2.08%</td>
<td>2.86%</td>
<td>4.35%</td>
<td>4.42%</td>
</tr>
<tr>
<td>8 bits (M=’00’), 2 Stop bit</td>
<td>2.08%</td>
<td>2.86%</td>
<td>4.35%</td>
<td>4.42%</td>
</tr>
<tr>
<td>9 bits (M=’01’), 2 Stop bit</td>
<td>1.82%</td>
<td>2.56%</td>
<td>3.90%</td>
<td>4.42%</td>
</tr>
<tr>
<td>7 bits (M=’10’), 2 Stop bit</td>
<td>2.34%</td>
<td>3.23%</td>
<td>4.92%</td>
<td>4.42%</td>
</tr>
</tbody>
</table>

Note: The data specified in Table 177 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = ’00’ (11-bit times when M=’01’ or 9-bit times when M = ’10’).

33.4.9 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all non addressed receivers.
The non addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_ker_ck cycles), otherwise Mute mode might remain active.

When the Mute mode is enabled:
- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in LPUART_ISR register is set to ‘1’. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The LPUART enters Mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of Mute mode behavior using Idle line detection is given in Figure 347.

Figure 347. Mute mode using Idle line detection

<table>
<thead>
<tr>
<th>RX</th>
<th>Data 1</th>
<th>Data 2</th>
<th>Data 3</th>
<th>Data 4</th>
<th>IDLE</th>
<th>Data 5</th>
<th>Data 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>RWU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MMRQ written to 1</td>
<td>Mute mode</td>
<td>Normal mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idle frame detected</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: If the MMRQ is set while the IDLE character has already elapsed, Mute mode will not be entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’ otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-
The bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the LPUART_CR2 register.

**Note:** In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters Mute mode.

The LPUART also enters Mute mode when the MMRQ bit is written to ‘1’. The RWU bit is also automatically set in this case.

The LPUART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

**Note:** When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in Mute mode.

An example of Mute mode behavior using address mark detection is given in Figure 348.

![Figure 348. Mute mode using address mark detection](image)

### 33.4.10 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in Table 178.

<table>
<thead>
<tr>
<th>M bits</th>
<th>PCE bit</th>
<th>LPUART frame(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit will be 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit will be 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

33.4.11 LPUART single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the LPUART_CR2 register,
- SCEN and IREN bits in the LPUART_CR3 register.

The LPUART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to ‘1’:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.
Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

**Note:** In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

### 33.4.12 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

**Note:** Refer to Section 32.4: USART implementation on page 953 to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 32.5.6. To perform continuous communication. When FIFO is disabled, you can clear the TXE/ RXNE flags in the LPUART_ISR register.

#### Transmission using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding Direct memory access controller section) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register.
5. Configure DMA interrupt generation after half/full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the LPUART communication is complete. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.
Figure 349. Transmission using DMA

<table>
<thead>
<tr>
<th>TX line</th>
<th>Frame 1</th>
<th>Frame 2</th>
<th>Frame 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXE flag</td>
<td>Set by hardware cleared by DMA read</td>
<td>Set by hardware cleared by DMA read</td>
<td>Set by hardware</td>
</tr>
<tr>
<td>DMA request</td>
<td></td>
<td></td>
<td>Ignored by the DMA because the transfer is complete</td>
</tr>
<tr>
<td>USART_TDR</td>
<td>F1</td>
<td>F2</td>
<td>F3</td>
</tr>
<tr>
<td>TC flag</td>
<td></td>
<td></td>
<td>Set by hardware</td>
</tr>
<tr>
<td>DMA writes USART_TDR</td>
<td></td>
<td>Cleared by software</td>
<td></td>
</tr>
<tr>
<td>DMA TCIF flag (transfer complete)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software configures DMA to send 3 data blocks and enables USART</td>
<td>DMA writes F1 into USART_TDR</td>
<td>DMA writes F2 into USART_TDR</td>
<td>DMA writes F3 into USART_TDR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The DMA transfer is complete (TCIF=1 in DMA_ISR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Software waits until TC=1</td>
</tr>
</tbody>
</table>

Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to the corresponding Direct memory access controller (DMA) section) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register.
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.
Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

33.4.13 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The Figure 337 shows how to connect 2 devices in this mode:
RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

**RS232 RTS flow control**

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 352* shows an example of communication with RTS flow control enabled.

![Figure 352. RS232 RTS flow control](image)

*Note:* When FIFO mode is enabled, nRTS is de-asserted only when RXFIFO is full.

**RS232 CTS flow control**

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 353* shows an example of communication with CTS flow control enabled.
Figure 353. RS232 CTS flow control

Note: For correct behavior, nCTS must be asserted at least 3 LPUART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This allows activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.

The LPUART DEAT and DEDT are expressed in LPUART clock source (fCK) cycles:

- The Driver enable assertion time equals
  - \((1 + (\text{DEAT} \times P)) \times f_{\text{CK}}, \text{if } P \neq 0\)
  - \((1 + \text{DEAT}) \times f_{\text{CK}}, \text{if } P = 0\)

- The Driver enable de-assertion time equals
  - \((1 + (\text{DEDT} \times P)) \times f_{\text{CK}}, \text{if } P \neq 0\)
  - \((1 + \text{DEDT}) \times f_{\text{CK}}, \text{if } P = 0\)

where \(P = BRR[20:11]\)
33.4.14 **LPUART low-power management**

The LPUART has advanced low-power mode functions allowing it to transfer properly data even when the luart_pclk clock is disabled.

The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the LPUART provides a wakeup interrupt (usart_wkup) if a specific action requiring the activation of the usart_pclk clock is needed:

- **If FIFO mode is disabled**
  - luart_pclk clock has to be activated to empty the LPUART data register.
  - In this case, the lpuart_wkup interrupt source is the RXNE set to ‘1’. The RXNEIE bit must be set before entering low-power mode.

- **If FIFO mode is enabled**
  - luart_pclk clock has to be activated
    - to fill the TXFIFO
    - or to empty the RXFIFO
  - In this case, the lpuart_wkup interrupt source can be:
    - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
    - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
    - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.
  - This allows sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the lpuart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wakeup time is less than the time to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode allows doing as many LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific lpuart_wkup interrupt may be selected through the WUS bitfields.

When the wakeup event is detected, the WUF flag is set by hardware and lpuart_wkup interrupt is generated if the WUFIE bit is set. In this case the lpuart_wkup interrupt is not mandatory for the wakeup. The WUF being set is sufficient to wakeup the MCU from low-power mode.
Note: Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in low-power or in an active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the LPUART is actually enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled upon exit from low-power mode.

When FIFO is enabled, the wakeup from low-power mode on address match is only possible when Mute mode is enabled.

Using Mute mode with low-power mode

If the LPUART is put into Mute mode before entering low-power mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wakeup from Mute mode on address match is used, then the low-power mode wakeup source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface will remain in Mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, Mute mode is used with wakeup from low-power mode without any constraints (i.e. the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wakeup from low-power mode when LPUART kernel clock lpuart_ker_ck is OFF in low-power mode

If during low-power mode, the lpuart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the lpuart_ker_ck clock to be switched ON thanks to the lpuart_ker_ck_req signal. The lpuart_ker_ck is then used for the frame reception.

If the wakeup event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wakeup event is not verified, the usart_ker_ck is switched OFF again, the MCU is not waken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of wakeup event programmed to “address match detection” and FIFO management disabled.

*Figure 354* shows the behavior when the wakeup event is verified.
Figure 354. Wakeup event verified (wakeup event = address match, FIFO disabled)

Figure 355. Wakeup event not verified (wakeup event = address match, FIFO disabled)

Note: The above figures are valid when address match or any received frame is used as wakeup event. In the case the wakeup event is the start bit detection, the LPUART sends the wakeup event to the MCU at the end of the start bit.
Determining the maximum LPUART baud rate that allows to correctly wake up the MCU from low-power mode

The maximum baud rate allowing to correctly wake up the MCU from low-power mode depends on the wakeup time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see Section 33.4.8: Tolerance of the LPUART receiver to clock deviation).

Let us take the example of OVER8 = 0, M bits = ‘01’, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to Table 177: Tolerance of the LPUART receiver, the LPUART receiver tolerance equals 3.41%.

\[
D_{WUmax} = \frac{t_{WULPUART}}{(11 \times T_{bit \ MIN})}
\]

\[
T_{bit \ MIN} = \frac{t_{WULPUART}}{(11 \times D_{WUmax})}
\]

where \( t_{WULPUART} \) is the wakeup time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

\[
t_{WULPUART} = 3 \ \mu s \ (values \ provided \ only \ as \ examples; \ for \ correct \ values, \ refer \ to \ the \ device \ datasheet).
\]

\[
D_{WUmax} = 3.41\% - 1\% = 2.41\%
\]

\[
T_{bit \ MIN} = 3 \ \mu s/ (11 \times 2.41\%) = 11.32 \ \mu s.
\]

As a result, the maximum baud rate allowing to wakeup correctly from low-power mode is: \( 1/11.32 \ \mu s = 88.36 \ \text{Kbaud} \).

### 33.5 LPUART interrupts

Refer to Table 172 for a detailed description of all LPUART interrupt requests.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Interrupt activated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit data register empty</td>
<td>TXE</td>
<td>TXEIE</td>
<td>TXE cleared when a data is written in TDR</td>
<td>YES \ NO</td>
</tr>
<tr>
<td>Transmit FIFO Not Full</td>
<td>TXFNF</td>
<td>TXFNFIE</td>
<td>TXFNF cleared when TXFIFO is full.</td>
<td>YES \ NO</td>
</tr>
<tr>
<td>Transmit FIFO Empty</td>
<td>TXFE</td>
<td>TXFEIE</td>
<td>TXFE cleared when the TXFIFO contains at least one data or by setting TXFRQ bit.</td>
<td>YES \ YES</td>
</tr>
<tr>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFTIE</td>
<td>TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold</td>
<td>YES \ YES</td>
</tr>
<tr>
<td>Interrupt event</td>
<td>Event flag</td>
<td>Enable Control bit</td>
<td>Interrupt clear method</td>
<td>Interrupt activated</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------------</td>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>CTS interrupt</td>
<td>CTSIF</td>
<td>CTSIE</td>
<td>CTSIF cleared by software by setting CTSCF bit.</td>
<td>YES</td>
</tr>
<tr>
<td>Transmission Complete</td>
<td>TC</td>
<td>TCIE</td>
<td>TC cleared when a data is written in TDR or by setting TCCF bit.</td>
<td>YES</td>
</tr>
<tr>
<td>Receive data register not empty (data ready to be read)</td>
<td>RXNE</td>
<td>RXNEIE</td>
<td>RXNE cleared by reading RDR or by setting RXFRQ bit.</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO Not Empty</td>
<td>RXFNE</td>
<td>RXFNEIE</td>
<td>RXFNE cleared when the RXFIFO is empty or by setting RXFRQ bit.</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO Full</td>
<td>RXFF(1)</td>
<td>RXFFIE</td>
<td>RXFF cleared when the RXFIFO contains at least one data.</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFTIE</td>
<td>RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold</td>
<td>YES YES</td>
</tr>
<tr>
<td>Overrun error detected</td>
<td>ORE</td>
<td>RX-NEIE/RX FNEIE</td>
<td>ORE cleared by setting ORECFC bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Idle line detected</td>
<td>IDLE</td>
<td>IDLEIE</td>
<td>IDLE cleared by setting IDLFCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Parity error</td>
<td>PE</td>
<td>PEIE</td>
<td>PE cleared by setting PECF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Noise Flag, Overrun error and Framing Error in multibuffer communication.</td>
<td>NE or ORE or FE</td>
<td>EIE</td>
<td>NE cleared by setting NECF bit. ORE cleared by setting ORECFC bit. FE flag cleared by setting FECF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Character match</td>
<td>CMF</td>
<td>CMIE</td>
<td>CMF cleared by setting CMCF bit.</td>
<td>YES NO</td>
</tr>
<tr>
<td>Wakeup from low-power mode</td>
<td>WUF</td>
<td>WUFIE</td>
<td>WUF is cleared by setting WUCFCF bit.</td>
<td>YES YES</td>
</tr>
<tr>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFTIE</td>
<td>TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold</td>
<td>YES YES</td>
</tr>
<tr>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFTIE</td>
<td>RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold</td>
<td>YES YES</td>
</tr>
</tbody>
</table>

1. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register will not be written and once n data are received and written in the RXFIFO, the RXFF interrupt will be asserted (RXFF flag is not set).
### 33.6 LPUART registers

Refer to Section 1.2 on page 50 for a list of abbreviations used in register descriptions.

#### 33.6.1 Control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00  
Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

**FIFO mode enabled**

<table>
<thead>
<tr>
<th>Bit 31 (RXFFIE)</th>
<th>RXFIFO Full interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>An LPUART interrupt is generated when RXFF=1 in the LPUART_ISR register</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30 (TXFEIE)</th>
<th>TXFIFO empty interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>An LPUART interrupt is generated when TXFE=1 in the LPUART_ISR register</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29 (FIFOEN)</th>
<th>FIFO mode enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FIFO mode is disabled.</td>
</tr>
<tr>
<td>1</td>
<td>FIFO mode is enabled.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 28 (M1)</th>
<th>Word length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1Start bit, 8 Data bits, n Stop bit</td>
</tr>
<tr>
<td>1</td>
<td>1Start bit, 9 Data bits, n Stop bit</td>
</tr>
</tbody>
</table>

This bit can only be written when the LPUART is disabled (UE=0).

**Note:** In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

<table>
<thead>
<tr>
<th>Bits 27:26</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 25:21</td>
<td>DEAT[4:0]: Driver Enable assertion time</td>
</tr>
</tbody>
</table>

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer Section 32.5.20: RS232 Hardware flow control and RS485 Driver Enable.

This bitfield can only be written when the LPUART is disabled (UE=0).
Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in \lpuart_ker_ck clock cycles. For more details, refer Section 33.4.13: RS232 Hardware flow control and RS485 Driver Enable.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited
1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the Mute mode function of the LPUART. When set, the LPUART can switch between the active and Mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently
1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE=0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the LPUART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line
1: Address mark

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software.

Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled
1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.

0: Even parity
1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever PE=1 in the LPUART_ISR register
Bit 7  **TXFNFIE**: TXFIFO not full interrupt enable
   This bit is set and cleared by software.
   0: Interrupt is inhibited
   1: A LPUART interrupt is generated whenever TXE/TXFNF =1 in the LPUART_ISR register

Bit 6  **TCIE**: Transmission complete interrupt enable
   This bit is set and cleared by software.
   0: Interrupt is inhibited
   1: An LPUART interrupt is generated whenever TC=1 in the LPUART_ISR register

Bit 5  **RXFNEIE**: RXFIFO not empty interrupt enable
   This bit is set and cleared by software.
   0: Interrupt is inhibited
   1: A LPUART interrupt is generated whenever ORE=1 or RXNE/RXFNE=1 in the LPUART_ISR register

Bit 4  **IDLEIE**: IDLE interrupt enable
   This bit is set and cleared by software.
   0: Interrupt is inhibited
   1: An LPUART interrupt is generated whenever IDLE=1 in the LPUART_ISR register

Bit 3  **TE**: Transmitter enable
   This bit enables the transmitter. It is set and cleared by software.
   0: Transmitter is disabled
   1: Transmitter is enabled

   **Note**: During transmission, a low pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

   *When TE is set there is a 1 bit-time delay before the transmission starts.*
Bit 2 **RE**: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in Stop mode
When this bit is cleared, the LPUART is not able to wake up the MCU from low-power mode.
When this bit is set, the LPUART is able to wake up the MCU from low-power mode, provided that the LPUART clock selection is HSI or LSE in the RCC.
This bit is set and cleared by software.
0: LPUART not able to wake up the MCU from low-power mode.
1: LPUART able to wake up the MCU from low-power mode. When this function is active, the clock source for the LPUART must be HSI or LSE (see RCC chapter)

**Note**: It is recommended to set the UESM bit just before entering low-power mode and clear it on exit from low-power mode.

Bit 0 **UE**: LPUART enable
When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.
0: LPUART prescaler and outputs disabled, low-power mode
1: LPUART enabled

**Note**: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

### 33.6.2 Control register 1 [alternate] (LPUART_CR1)
Address offset: 0x00
Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

#### FIFO mode disabled

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Res.</th>
<th>CMIE</th>
<th>MME</th>
<th>M0</th>
<th>WAKE</th>
<th>PCE</th>
<th>PS</th>
<th>PEIE</th>
<th>TXEIE</th>
<th>TCIE</th>
<th>RXNEIE</th>
<th>IDLEIE</th>
<th>TE</th>
<th>RE</th>
<th>UESM</th>
<th>UE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:30  Reserved, must be kept at reset value.

Bit 29  **FIFOEN**: FIFO mode enable
   This bit is set and cleared by software.
   0: FIFO mode is disabled.
   1: FIFO mode is enabled.

Bit 28  **M1**: Word length
   This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
   \[M[1:0] = '00' \text{: 1 Start bit, 8 Data bits, n Stop bit}\]
   \[M[1:0] = '01' \text{: 1 Start bit, 9 Data bits, n Stop bit}\]
   \[M[1:0] = '10' \text{: 1 Start bit, 7 Data bits, n Stop bit}\]
   This bit can only be written when the LPUART is disabled (UE=0).
   **Note:** In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26  Reserved, must be kept at reset value.

Bits 25:21  **DEAT[4:0]**: Driver Enable assertion time
   This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer Section 32.5.20: RS232 Hardware flow control and RS485 Driver Enable.
   This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 20:16  **DEDT[4:0]**: Driver Enable deassertion time
   This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in lpuart_ker_ck clock cycles. For more details, refer Section 33.4.13: RS232 Hardware flow control and RS485 Driver Enable.
   If the LPUART TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.
   This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15  Reserved, must be kept at reset value.

Bit 14  **CMIE**: Character match interrupt enable
   This bit is set and cleared by software.
   0: Interrupt is inhibited
   1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13  **MME**: Mute mode enable
   This bit activates the Mute mode function of the LPUART. When set, the LPUART can switch between the active and Mute modes, as defined by the WAKE bit. It is set and cleared by software.
   0: Receiver in active mode permanently
   1: Receiver can switch between Mute mode and active mode.

Bit 12  **M0**: Word length
   This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).
   This bit can only be written when the LPUART is disabled (UE=0).
Bit 11 **WAKE**: Receiver wakeup method

This bit determines the LPUART wakeup method from Mute mode. It is set or cleared by software.

- 0: Idle line
- 1: Address mark

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software.

- 0: Parity control disabled
- 1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.

- 0: Even parity
- 1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever PE=1 in the LPUART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated whenever TXE/TXFNF =1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever TC=1 in the LPUART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated whenever ORE=1 or RXNE/RXFNE=1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever IDLE=1 in the LPUART_ISR register
Bit 3  **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled
1: Transmitter is enabled

**Note:** During transmission, a low pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2  **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1  **UESM**: LPUART enable in Stop mode

When this bit is cleared, the LPUART is not able to wake up the MCU from low-power mode.

When this bit is set, the LPUART is able to wake up the MCU from low-power mode, provided that the LPUART clock selection is HSI or LSE in the RCC.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.
1: LPUART able to wake up the MCU from low-power mode. When this function is active, the clock source for the LPUART must be HSI or LSE (see RCC chapter)

**Note:** It is recommended to set the UESM bit just before entering low-power mode and clear it on exit from low-power mode.

Bit 0  **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode
1: LPUART enabled

**Note:** To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

### 33.6.3 Control register 2 (LPUART_CR2)

**Address offset:** 0x04

**Reset value:** 0x0000 0000

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

1074/1234  RM0444 Rev 2
Bits 31:24  **ADD[7:0]:** Address of the LPUART node

**ADD[7:4]:**
These bits give the address of the LPUART node or a character code to be recognized. They are used to wake up the MCU with 7-bit address mark detection in multiprocessor communication during Mute mode or Stop mode. The MSB of the character sent by the transmitter should be equal to 1. They can also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match. These bits can only be written when reception is disabled (RE = 0) or the LPUART is disabled (UE=0)

**ADD[3:0]:**
These bits give the address of the LPUART node or a character code to be recognized. They are used for wakeup with address mark detection in multiprocessor communication during Mute mode or low-power mode. These bits can only be written when reception is disabled (RE = 0) or the LPUART is disabled (UE=0)

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

**Bit 19  MSBFIRST:** Most significant bit first
This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bitfield can only be written when the LPUART is disabled (UE=0).

**Bit 18  DATAINV:** Binary data inversion
This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bitfield can only be written when the LPUART is disabled (UE=0).

**Bit 17  TXINV:** TX pin active level inversion
This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (VDD =1/idle, Gnd=0/mark)
1: TX pin signal values are inverted. ((VDD =0/mark, Gnd=1/idle).
This allows the use of an external inverter on the TX line.
This bitfield can only be written when the LPUART is disabled (UE=0).

**Bit 16  RXINV:** RX pin active level inversion
This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (VDD =1/idle, Gnd=0/mark)
1: RX pin signal values are inverted. ((VDD =0/mark, Gnd=1/idle).
This allows the use of an external inverter on the RX line.
This bitfield can only be written when the LPUART is disabled (UE=0).

**Bit 15  SWAP:** Swap TX/RX pins
This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This allows to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the LPUART is disabled (UE=0).

**Bit 14  Reserved, must be kept at reset value.**
33.6.4 Control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DEP</td>
<td>DEM</td>
<td>DDRE</td>
<td>OVRD</td>
<td>S</td>
<td>Res</td>
<td>CTSIE</td>
<td>CTSE</td>
<td>RTSE</td>
<td>DMAT</td>
<td>DMAR</td>
<td>Res</td>
<td>Res</td>
<td>HDSEL</td>
<td>Res</td>
<td>Res</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 13:12 STOP[1:0]: STOP bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 ADDM7:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE=0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address
(ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.
Bits 31:29 **TXFTCFG[2:0]**: TXFIFO threshold configuration
- 000: TXFIFO reaches 1/8 of its depth.
- 001: TXFIFO reaches 1/4 of its depth.
- 110: TXFIFO reaches 1/2 of its depth.
- 100: TXFIFO reaches 7/8 of its depth.
- 101: TXFIFO becomes empty.
- Remaining combinations: Reserved.

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]**: Receive FIFO threshold configuration
- 000: Receive FIFO reaches 1/8 of its depth.
- 001: Receive FIFO reaches 1/4 of its depth.
- 110: Receive FIFO reaches 1/2 of its depth.
- 011: Receive FIFO reaches 3/4 of its depth.
- 100: Receive FIFO reaches 7/8 of its depth.
- 101: Receive FIFO becomes full.
- Remaining combinations: Reserved.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wakeup from low-power mode interrupt enable
This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever WUF=1 in the LPUART_ISR register

*Note:* **WUFIE must be set before entering in low-power mode.**

If the LPUART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation.

Bits 21:20 **WUS[1:0]**: Wakeup from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (Wakeup from low-power mode flag).
- 00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
- 01: Reserved.
- 10: WUF active on Start bit detection
- 11: WUF active on RXNE.

This bitfield can only be written when the LPUART is disabled (UE=0).

*Note:* If the LPUART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 32.4: USART implementation.

Bits 19:16 Reserved, must be kept at reset value.
Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.
1: DE signal is active low.

This bit can only be written when the LPUART is disabled (UE=0).

Bit 14 **DEM**: Driver enable mode

This bit allows the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the LPUART is disabled (UE=0).

Bit 13 **DDRE**: DMA Disable on Reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data will be transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.

This bit can only be written when the LPUART is disabled (UE=0).

**Note:** The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.

This bit can only be written when the LPUART is disabled (UE=0).

**Note:** This control bit allows checking the communication flow w/o reading the data.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF=1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0).

If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.

This bit can only be written when the LPUART is disabled (UE=0)

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.

This bit can only be written when the LPUART is disabled (UE=0).

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software
0: DMA mode is disabled for transmission
1: DMA mode is enabled for transmission
Bit 6 **DMAR:** DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL:** Half-duplex selection
Selection of Single-wire Half-duplex mode
0: Half duplex mode is not selected
1: Half duplex mode is selected
This bit can only be written when the LPUART is disabled (UE=0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE:** Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NE=1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE=1 or ORE=1 or NE=1 in the LPUART_ISR register.

### 33.6.5 Baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
<tr>
<td>15</td>
<td>BRR[19:16]</td>
<td>LPUART baud rate</td>
</tr>
<tr>
<td>14</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>13</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>12</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>11</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>10</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>9</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>8</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>7</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>6</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>5</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>4</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>3</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>2</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>1</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
<tr>
<td>0</td>
<td>rw</td>
<td>Readable and writable</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]:** LPUART baud rate

**Note:**

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be ≥ 0x300 and LPUART_BRR is 20 bits, a care should be taken when generating high baud rates using high fck values. fck must be in the range [3 x baud rate..4096 x baud rate].
33.6.6 Request register (LPUART_RQR)

Address offset: 0x18
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>4</td>
<td>TXFRQ: Transmit data flush request</td>
</tr>
<tr>
<td></td>
<td>This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This will set the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).</td>
</tr>
<tr>
<td></td>
<td>Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.</td>
</tr>
<tr>
<td>3</td>
<td>RXFRQ: Receive data flush request</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the RXNE flag. This allows discarding the received data without reading it, and avoid an overrun condition.</td>
</tr>
<tr>
<td>2</td>
<td>MMRQ: Mute mode request</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.</td>
</tr>
<tr>
<td>1</td>
<td>SBKRQ: Send break request</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.</td>
</tr>
<tr>
<td></td>
<td>Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.</td>
</tr>
<tr>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

33.6.7 Interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C
Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>4</td>
<td>TXFRQ: Transmit data flush request</td>
</tr>
<tr>
<td></td>
<td>This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This will set the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).</td>
</tr>
<tr>
<td></td>
<td>Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.</td>
</tr>
<tr>
<td>3</td>
<td>RXFRQ: Receive data flush request</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the RXNE flag. This allows discarding the received data without reading it, and avoid an overrun condition.</td>
</tr>
<tr>
<td>2</td>
<td>MMRQ: Mute mode request</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.</td>
</tr>
<tr>
<td>1</td>
<td>SBKRQ: Send break request</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.</td>
</tr>
<tr>
<td></td>
<td>Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.</td>
</tr>
<tr>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

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

Bit 27 **TXFT**: TXFIFO threshold flag
This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG in LPUART_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit =1 (bit 31) in the LPUART_CR3 register.
0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag
This bit is set by hardware when the RXFIFO reaches the threshold programmed in RXFTCFG in LPUART_CR3 register i.e. the Receive FIFO contains RXFTCFG data. An interrupt is generated if the RXFTIE bit =1 (bit 27) in the LPUART_CR3 register.
0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO full
This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register. An interrupt is generated if the RXFFIE bit =1 in the LPUART_CR1 register.
0: RXFIFO is not full
1: RXFIFO is full

Bit 23 **TXFE**: TXFIFO empty
This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register. An interrupt is generated if the TXFEIE bit =1 (bit 30) in the LPUART_CR1 register.
0: TXFIFO is not empty
1: TXFIFO is empty

Bit 22 **REACK**: Receive enable acknowledge flag
This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.
It can be used to verify that the LPUART is ready for reception before entering low-power mode.
Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag
This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.
It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the LPUART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag
This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register. An interrupt is generated if WUFIE=1 in the LPUART_CR3 register.
Note: When UESM is cleared, WUF flag is also cleared.
If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.
Bit 19  **RWU**: Receiver wakeup from Mute mode

This bit indicates if the LPUART is in Mute mode. It is cleared/set by hardware when a wakeuup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in Active mode
1: Receiver in Mute mode

**Note:** If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.

Bit 18  **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character is transmitted
1: Break character will be transmitted

Bit 17  **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE=1 in the LPUART_CR1 register.

0: No Character match detected
1: Character Match detected

Bit 16  **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)
1: Reception on going

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

Bit 10  **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set
1: nCTS line reset

**Note:** If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9  **CTSIF**: CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE=1 in the LPUART_CR3 register.

0: No change occurred on the nCTS status line
1: A change occurred on the nCTS status line

**Note:** If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8  Reserved, must be kept at reset value.
Bit 7 TXFNF: TXFIFO not full
TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE will be set at the same time).

An interrupt is generated if the TXFNFIE bit = 1 in the LPUART_CR1 register.
0: Data register is full/Transmit FIFO is full.
1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 TC: Transmission complete
This bit is set by hardware if the transmission of a frame containing data is complete and if TXFF is set. An interrupt is generated if TCIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the TCCF in the LPUART_ICR register or by a write to the LPUART_TDR register.

An interrupt is generated if TCIE=1 in the LPUART_CR1 register.
0: Transmission is not complete
1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit will be set immediately.

Bit 5 RXFNE: RXFIFO not empty
RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFREQ in the LPUART_RQR register.

An interrupt is generated if RXFNEIE=1 in the LPUART_CR1 register.
0: Data is not received
1: Received data is ready to be read.

Bit 4 IDLE: Idle line detected
This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected
1: Idle line is detected

Note: The IDLE bit will not be set again until the RXFNE bit has been set (i.e. a new idle line occurs).

If Mute mode is enabled (MME=1), IDLE is set if the LPUART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.
Bit 3  **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXFNEIE=1 or EIE = 1 in the LPUART_CR1 register.

0: No overrun error
1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2  **NE**: Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NECF bit in the LPUART_ICR register.

0: No noise is detected
1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

This error is associated with the character in the LPUART_RDR.

Bit 1  **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR1 register.

0: No Framing error is detected
1: Framing error or break character is detected

Note: This error is associated with the character in the LPUART_RDR.

Bit 0  **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error
1: Parity error

Note: This error is associated with the character in the LPUART_RDR.
## 33.6.8 Interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

### FIFO mode disabled

<table>
<thead>
<tr>
<th>Bit 31:23 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 22 <strong>REACK</strong>: Receive enable acknowledge flag</td>
</tr>
<tr>
<td>This bit is set/reset by hardware when the Receive Enable value is taken into account by the LPUART.</td>
</tr>
<tr>
<td>It can be used to verify that the LPUART is ready for reception before entering low-power mode.</td>
</tr>
<tr>
<td><strong>Note:</strong> If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.</td>
</tr>
<tr>
<td>Bit 21 <strong>TEACK</strong>: Transmit enable acknowledge flag</td>
</tr>
<tr>
<td>This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.</td>
</tr>
<tr>
<td>It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the LPUART_CR1 register, in order to respect the TE=0 minimum period.</td>
</tr>
<tr>
<td>Bit 20 <strong>WUF</strong>: Wakeup from low-power mode flag</td>
</tr>
<tr>
<td>This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register. An interrupt is generated if WUFIE=1 in the LPUART_CR3 register.</td>
</tr>
<tr>
<td><strong>Note:</strong> When UESM is cleared, WUF flag is also cleared.</td>
</tr>
<tr>
<td>If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.</td>
</tr>
<tr>
<td>Bit 19 <strong>RWU</strong>: Receiver wakeup from Mute mode</td>
</tr>
<tr>
<td>This bit indicates if the LPUART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.</td>
</tr>
<tr>
<td>When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.</td>
</tr>
<tr>
<td>0: Receiver in active mode</td>
</tr>
<tr>
<td>1: Receiver in Mute mode</td>
</tr>
<tr>
<td><strong>Note:</strong> If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.</td>
</tr>
</tbody>
</table>
Bit 18 **SBKF**: Send break flag
This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.
0: No break character is transmitted
1: Break character will be transmitted

Bit 17 **CMF**: Character match flag
This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.
An interrupt is generated if CMIE=1 in the LPUART_CR1 register.
0: No Character match detected
1: Character Match detected

Bit 16 **BUSY**: Busy flag
This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).
0: LPUART is idle (no reception)
1: Reception on going

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

Bit 10 **CTS**: CTS flag
This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.
0: nCTS line set
1: nCTS line reset
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag
This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.
An interrupt is generated if CTSIE=1 in the LPUART_CR3 register.
0: No change occurred on the nCTS status line
1: A change occurred on the nCTS status line
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE**: Transmit data register empty/TXFIFO not full
TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.
An interrupt is generated if the TXEIE bit =1 in the LPUART_CR1 register.
0: Data register full
1: Data register not full
*Note: This bit is used during single buffer transmission.*
Bit 6  **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE is set. An interrupt is generated if TCIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the TCCF in the LPUART_ICR register or by a write to the LPUART_TDR register.

An interrupt is generated if TCIE=1 in the LPUART_CR1 register.

0: Transmission is not complete
1: Transmission is complete

*Note: If TE bit is reset and no transmission is on going, the TC bit will be set immediately.*

Bit 5  **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by reading from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE=1 in the LPUART_CR1 register.

0: Data is not received
1: Received data is ready to be read.

Bit 4  **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected
1: Idle line is detected

*Note: The IDLE bit will not be set again until the RXNE bit has been set (i.e. a new idle line occurs).*

If Mute mode is enabled (MME=1), IDLE is set if the LPUART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3  **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE=1. It is cleared by software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXNEIE=1 or EIE = 1 in the LPUART_CR1 register.

0: No overrun error
1: Overrun error is detected

*Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.
Bit 2 **NE**: Start bit noise detection flag
This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NECF bit in the LPUART_ICR register.
0: No noise is detected
1: Noise is detected
*Note:* This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

Bit 1 **FE**: Framing error
This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.
When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the LPUART_CR1 register.
0: No Framing error is detected
1: Framing error or break character is detected

Bit 0 **PE**: Parity error
This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.
An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.
0: No parity error
1: Parity error

### 33.6.9 Interrupt flag clear register (LPUART_ICR)
Address offset: 0x20
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td><strong>WUCF</strong>: Wakeup from low-power mode clear flag</td>
</tr>
<tr>
<td>Note:</td>
<td>If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 32.4: USART implementation.</td>
</tr>
<tr>
<td>19-18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td><strong>CMCF</strong>: Character match clear flag</td>
</tr>
<tr>
<td>Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.</td>
<td></td>
</tr>
<tr>
<td>16-10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>9</td>
<td><strong>CTSCF</strong>: CTS clear flag</td>
</tr>
<tr>
<td>Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.</td>
<td></td>
</tr>
<tr>
<td>8-7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
33.6.10 Receive data register (LPUART_RDR)

Address offset: 0x24
Reset value: 0x0000 0000

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

Bits 31:9  Reserved, must be kept at reset value.

Bits 8:0  **RDR[8:0]**: Receive data value

Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see Figure 342). When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

33.6.11 Transmit data register (LPUART_TDR)

Address offset: 0x28
Reset value: 0x0000 0000

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

Bits 31:9  Reserved, must be kept at reset value.

Bits 8:0  **TDR[8:0]**: Transmit data value

Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 5  Reserved, must be kept at reset value.

Bit 4  **IDLECF**: Idle line detected clear flag

Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 3  **ORECF**: Overrun error clear flag

Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2  **NECF**: Noise detected clear flag

Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1  **FECF**: Framing error clear flag

Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0  **PECF**: Parity error clear flag

Writing 1 to this bit clears the PE flag in the LPUART_ISR register.
33.6.12 Prescaler register (LPUART_PRES)

This register can only be written when the LPUART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000 0000

| Bits 31:9 | Reserved, must be kept at reset value. |
| Bits 8:0 | **TDR[8:0]**: Transmit data value |
| Contains the data character to be transmitted. |
| The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 342). |
| When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity. |

**Note:** This register must be written only when TXE/TXFNF=1.

| Bits 5:0 | PRESCALER[3:0]: Clock prescaler |
| The LPUART input clock can be divided by a prescaler: |
| 0000: input clock not divided |
| 0001: input clock divided by 2 |
| 0010: input clock divided by 4 |
| 0011: input clock divided by 6 |
| 0100: input clock divided by 8 |
| 0101: input clock divided by 10 |
| 0110: input clock divided by 12 |
| 0111: input clock divided by 16 |
| 1000: input clock divided by 32 |
| 1001: input clock divided by 64 |
| 1010: input clock divided by 128 |
| 1011: input clock divided by 256 |
| Remaining combinations: Reserved. |

**Note:** When **PRESCALER** is programmed with a value different of the allowed ones, programmed prescaler value will be «1011» i.e. input clock divided by 256.
### 33.6.13 LPUART register map

The table below gives the LPUART register map and reset values.

**Table 180. LPUART register map and reset values**

| Offset | Register name           | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|-------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | LPUART_CR1              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | FIFO mode enabled       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x00   | LPUART_CR1              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | FIFO mode disabled      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x04   | LPUART_CR2              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | ADD[7:0]                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x08   | LPUART_CR3              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TXFCFG[2:0]             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | RXFCFG[2:0]             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0C   | LPUART_BRR              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | BRR[19:0]               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x10   | Reserved                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10-  | Reserved                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x18   | LPUART_ROR             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1C   | LPUART_ISR             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | FIFO mode enabled       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1C   | LPUART_ISR             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | FIFO mode disabled      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
Refer to Section 2.2: Memory organization for the register boundary addresses.

| Offset | Register name     | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|        | Reset value      | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
34 Serial peripheral interface / inter-IC sound (SPI/I2S)

34.1 Introduction

The SPI/I²S interface can be used to communicate with external devices using the SPI protocol or the I²S audio protocol. SPI or I²S mode is selectable by software. SPI Motorola mode is selected by default after a device reset.

The serial peripheral interface (SPI) protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

The Inter-IC sound (I²S) protocol is also a synchronous serial communication interface. It can operate in slave or master mode with half-duplex communication. It can address four different audio standards including the Philips I²S standard, the MSB- and LSB-justified standards and the PCM standard.

34.2 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4 to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to \( f_{PCLK}/2 \)
- Slave mode frequency up to \( f_{PCLK}/2 \).
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI Motorola support
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun flags with interrupt capability
- CRC Error flag
- Two 32-bit embedded Rx and Tx FIFOs with DMA capability
- Enhanced TI and NSS pulse modes support
34.3 **I2S main features**

- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler to reach accurate audio sample frequencies (from 8 kHz to 192 kHz)
- Data format may be 16-bit, 24-bit or 32-bit
- Packet frame is fixed to 16-bit (16-bit data frame) or 32-bit (16-bit, 24-bit, 32-bit data frame) by audio channel
- Programmable clock polarity (steady state)
- Underrun flag in slave transmission mode, overrun flag in reception mode (master and slave) and Frame Error Flag in reception and transmitter mode (slave only)
- 16-bit register for transmission and reception with one data register for both channel sides
- Supported I2S protocols:
  - I2S Philips standard
  - MSB-justified standard (left-justified)
  - LSB-justified standard (right-justified)
  - PCM standard (with short and long frame synchronization on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first
- DMA capability for transmission and reception (16-bit wide)
- Master clock can be output to drive an external audio component. Ratio is fixed at $256 \times F_S$ (where $F_S$ is the audio sampling frequency)

34.4 **SPI/I2S implementation**

The following table describes all the SPI instances and their features embedded in the devices.

<table>
<thead>
<tr>
<th>SPI Features</th>
<th>SPI2S1</th>
<th>SPI2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhanced NSSP &amp; TI modes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>I2S support</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Hardware CRC calculation</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Data size configuration</td>
<td>from 4 to 16-bit</td>
<td>from 4 to 16-bit</td>
</tr>
<tr>
<td>Rx/Tx FIFO size</td>
<td>32-bit</td>
<td>32-bit</td>
</tr>
<tr>
<td>Wakeup capability from Low-power Sleep</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
34.5 SPI functional description

34.5.1 General description

The SPI allows synchronous, serial communication between the MCU and external devices. Application software can manage the communication by polling the status flag or using dedicated SPI interrupt. The main elements of SPI and their interactions are shown in the following block diagram Figure 356.

Figure 356. SPI block diagram

Four I/O pins are dedicated to SPI communication with external devices.

- **MISO**: Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI**: Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK**: Serial Clock output pin for SPI masters and input pin for SPI slaves.
- **NSS**: Slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - select an individual slave device for communication
  - synchronize the data frame or
  - detect a conflict between multiple masters

See Section 34.5.5: Slave select (NSS) pin management for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires - one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management.
34.5.2 Communications between one master and one slave

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software NSS management) or 3 or 4 wires (with hardware NSS management). Communication is always initiated by the master.

Full-duplex communication

By default, the SPI is configured for full-duplex communication. In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During SPI communication, data is shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

![Figure 357. Full-duplex single master/ single slave application](image)

1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see Section 34.5.5: Slave select (NSS) pin management.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting the BIDIMODE bit in the SPIx_CR1 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data is synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the BDIOE bit in their SPIx_CR1 registers. In this configuration, the master’s MISO pin and the slave’s MOSI pin are free for other application uses and act as GPIOs.
Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the RXONLY bit in the SPIx_CR2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO and MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode (RXONLY=0)**: The configuration settings are the same as for full-duplex. The application has to ignore the information captured on the unused input pin. This pin can be used as a standard GPIO.

- **Receive-only mode (RXONLY=1)**: The application can disable the SPI output function by setting the RXONLY bit. In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see 34.5.5: Slave select (NSS) pin management). Received data events appear depending on the data buffer configuration. In the master configuration, the MOSI output is disabled and the pin can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled. The only way to stop the clock is to clear the RXONLY bit or the SPE bit and wait until the incoming pattern from the MISO pin is finished and fills the data buffer structure, depending on its configuration.
Figure 359. Simplex single master/single slave application (master in transmit-only/slave in receive-only mode)

1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see Section 34.5.5: Slave select (NSS) pin management.

2. An accidental input information is captured at the input of transmitter Rx shift register. All the events associated with the transmitter receive flow must be ignored in standard transmit only mode (e.g. OVF flag).

3. In this configuration, both the MISO pins can be used as GPIOs.

Note: Any simplex communication can be alternatively replaced by a variant of the half-duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled while BDIO bit is not changed).

34.5.3 Standard multi-slave communication

In a configuration with two or more independent slaves, the master uses GPIO pins to manage the chip select lines for each slave (see Figure 360.). The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input. When this is done, a standard master and dedicated slave communication is established.
1. NSS pin is not used on master side at this configuration. It has to be managed internally (SSM=1, SSI=1) to prevent any MODF error.

2. As MISO pins of the slaves are connected together, all slaves must have the GPIO configuration of their MISO pin set as alternate function open-drain (see Section 6.3.7: I/O alternate function input/output).

### 34.5.4 Multi-master communication

Unless SPI bus is not designed for a multi-master capability primarily, the user can use build-in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, NSS pin is used configured at hardware input mode.

The connection of more than two SPI nodes working at this mode is impossible as only one node can apply its output on a common data line at time.

When nodes are non active, both stay at slave mode by default. Once one node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via dedicated GPIO pin. After the session is completed, the active slave select signal is released and the node mastering the bus temporary returns back to passive slave mode waiting for next session start.
If potentially both nodes raised their mastering request at the same time a bus conflict event appears (see mode fault MODF event). Then the user can apply some simple arbitration process (e.g. to postpone next attempt by predefined different time-outs applied at both nodes).

Figure 361. Multi-master application

1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as the passive node is configured as a slave.

34.5.5 Slave select (NSS) pin management

In slave mode, the NSS works as a standard “chip select” input and lets the slave communicate with the master. In master mode, NSS can be used either as output or input. As an input it can prevent multimaster bus collision, and as an output it can drive a slave select signal of a single slave.

Hardware or software slave select management can be set using the SSM bit in the SPIx_CR1 register:

- **Software NSS management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in register SPIx_CR1. The external NSS pin is free for other application uses.

- **Hardware NSS management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in register SPIx_CR1).
  - **NSS output enable (SSM=0, SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The NSS signal is driven low as soon as the SPI is enabled in master mode (SPE=1), and is kept low until the SPI is disabled (SPE =0). A pulse can be generated between continuous communications if NSS pulse mode is activated (NSSP=1). The SPI cannot work in multimaster configuration with this NSS setting.
  - **NSS output disable (SSM=0, SSOE = 0)**: if the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled low in this mode, the SPI enters master mode fault state and the device is automatically reconfigured in slave mode. In slave mode, the NSS pin works as a standard “chip select” input and the slave is selected while NSS line is at low level.
34.5.6 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slaves devices must follow the same communication format.

Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPIx_CR1 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.
Figure 363, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

**Note:** Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit. The idle state of SCK must correspond to the polarity selected in the SPIx_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

### Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFIRST bit. The data frame size is chosen by using the DS bits. It can be set from 4-bit up to 16-bit length and the setting applies for both transmission and reception. Whatever the selected data frame size, read access to the FIFO must be aligned with the FRXTH level. When the SPIx_DR register is accessed, data frames are always right-aligned into either a byte (if the data fits into a byte) or a half-word (see Figure 364). During communication, only bits within the data frame are clocked and transferred.
Note: The minimum data length is 4 bits. If a data length of less than 4 bits is selected, it is forced to an 8-bit data frame size.

34.5.7 Configuration of SPI

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated sections. When a standard communication is to be initialized, perform these steps:

1. Write proper GPIO registers: Configure GPIO for MOSI, MISO and SCK pins.
2. Write to the SPI_CR1 register:
   a) Configure the serial clock baud rate using the BR[2:0] bits (Note: 4).
   b) Configure the CPOL and CPHA bits combination to define one of the four relationships between the data transfer and the serial clock (CPHA must be cleared in NSSP mode). (Note: 2 - except the case when CRC is enabled at TI mode).
   c) Select simplex or half-duplex mode by configuring RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE can't be set at the same time).
   d) Configure the LSBFIRST bit to define the frame format (Note: 2).
   e) Configure the CRCL and CRCEN bits if CRC is needed (while SCK clock signal is at idle state).
   f) Configure SSM and SSI (Notes: 2 & 3).
   g) Configure the MSTR bit (in multimaster NSS configuration, avoid conflict state on NSS if master is configured to prevent MODF error).
3. Write to SPI_CR2 register:
   a) Configure the DS[3:0] bits to select the data length for the transfer.
   b) Configure SSOE (Notes: 1 & 2 & 3).
   c) Set the FRF bit if the TI protocol is required (keep NSSP bit cleared in TI mode).
   d) Set the NSSP bit if the NSS pulse mode between two data units is required (keep CHPA and TI bits cleared in NSSP mode).
   e) Configure the FRXTH bit. The RXFIFO threshold must be aligned to the read access size for the SPIx_DR register.
   f) Initialize LDMA_TX and LDMA_RX bits if DMA is used in packed mode.
4. Write to SPI_CRCPR register: Configure the CRC polynomial if needed.
5. Write proper DMA registers: Configure DMA streams dedicated for SPI Tx and Rx in DMA registers if the DMA streams are used.
**Note:**

(1) Step is not required in slave mode.
(2) Step is not required in TI mode.
(3) Step is not required in NSSP mode.
(4) The step is not required in slave mode except slave working at TI mode

### 34.5.8 Procedure for enabling SPI

It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave must already contain data to be sent before starting communication with the master (either on the first edge of the communication clock, or before the end of the ongoing communication if the clock signal is continuous). The SCK signal must be settled at an idle state level corresponding to the selected polarity before the SPI slave is enabled.

The master at full-duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled and TXFIFO is not empty, or with the next write to TXFIFO.

In any master receive only mode (RXONLY=1 or BIDIMODE=1 and BIDIOE=0), master starts to communicate and the clock starts running immediately after SPI is enabled.

For handling DMA, follow the dedicated section.

### 34.5.9 Data transmission and reception procedures

#### RXFIFO and TXFIFO

All SPI data transactions pass through the 32-bit embedded FIFOs. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short. Each direction has its own FIFO called TXFIFO and RXFIFO. These FIFOs are used in all SPI modes except for receiver-only mode (slave or master) with CRC calculation enabled (see Section 34.5.14: CRC calculation).

The handling of FIFOs depends on the data exchange mode (duplex, simplex), data frame format (number of bits in the frame), access size performed on the FIFO data registers (8-bit or 16-bit), and whether or not data packing is used when accessing the FIFOs (see Section 34.5.13: TI mode).

A read access to the SPIx_DR register returns the oldest value stored in RXFIFO that has not been read yet. A write access to the SPIx_DR stores the written data in the TXFIFO at the end of a send queue. The read access must be always aligned with the RXFIFO threshold configured by the FRXTH bit in SPIx_CR2 register. FTLVL[1:0] and FRLVL[1:0] bits indicate the current occupancy level of both FIFOs.

A read access to the SPIx_DR register must be managed by the RXNE event. This event is triggered when data is stored in RXFIFO and the threshold (defined by FRXTH bit) is reached. When RXNE is cleared, RXFIFO is considered to be empty. In a similar way, write access of a data frame to be transmitted is managed by the TXE event. This event is triggered when the TXFIFO level is less than or equal to half of its capacity. Otherwise TXE is cleared and the TXFIFO is considered as full. In this way, TXFIFO can only store up to three when the data frame format is not greater than 8 bits. This difference prevents possible corruption of 3x 8-bit data frames already stored in the TXFIFO when software tries to write more data in 16-bit mode into TXFIFO. Both TXE and RXNE events can be polled or handled by interrupts. See Figure 366 through Figure 369.
Another way to manage the data exchange is to use DMA (see Communication using DMA (direct memory addressing)).

If the next data is received when the RXFIFO is full, an overrun event occurs (see description of OVR flag at Section 34.5.10: SPI status flags). An overrun event can be polled or handled by an interrupt.

The BSY bit being set indicates ongoing transaction of a current data frame. When the clock signal runs continuously, the BSY flag stays set between data frames at master but becomes low for a minimum duration of one SPI clock at slave between each data frame transfer.

**Sequence handling**

A few data frames can be passed at single sequence to complete a message. When transmission is enabled, a sequence begins and continues while any data is present in the TXFIFO of the master. The clock signal is provided continuously by the master until TXFIFO becomes empty, then it stops waiting for additional data.

In receive-only modes, half-duplex (BIDIMODE=1, BIDIOE=0) or simplex (BIDIMODE=0, RXONLY=1) the master starts the sequence immediately when both SPI is enabled and receive-only mode is activated. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled by the master. The master receives data frames continuously up to this moment.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays. Be aware there is no underflow error signal for master or slave in SPI mode, and data from the slave is always transacted and processed by the master even if the slave could not prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are shorter and bus rate is high.

Each sequence must be encased by the NSS pulse in parallel with the multislave system to select just one of the slaves for communication. In a single slave system it is not necessary to control the slave with NSS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. NSS can be managed by both software and hardware (see Section 34.5.5: Slave select (NSS) pin management).

When the BSY bit is set it signifies an ongoing data frame transaction. When the dedicated frame transaction is finished, the RXNE flag is raised. The last bit is just sampled and the complete data frame is stored in the RXFIFO.

**Procedure for disabling the SPI**

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph. It is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Ongoing transactions can be corrupted in this case. In some modes the disable procedure is the only way to stop continuous communication running.

Master in full-duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. Special care must be taken in packing mode when an odd number of data frames are transacted to prevent some dummy byte exchange (refer to Data packing section). Before the SPI is disabled in these modes, the user must follow standard disable procedure. When
the SPI is disabled at the master transmitter while a frame transaction is ongoing or next data frame is stored in TXFIFO, the SPI behavior is not guaranteed.

When the master is in any receive only mode, the only way to stop the continuous clock is to disable the peripheral by SPE=0. This must occur in specific time window within last data frame transaction just between the sampling time of its first bit and before its last bit transfer starts (in order to receive a complete number of expected data frames and to prevent any additional “dummy” data reading after the last valid data frame). Specific procedure must be followed when disabling SPI in this mode.

Data received but not read remains stored in RXFIFO when the SPI is disabled, and must be processed the next time the SPI is enabled, before starting a new sequence. To prevent having unread data, ensure that RXFIFO is empty when disabling the SPI, by using the correct disabling procedure, or by initializing all the SPI registers with a software reset via the control of a specific register dedicated to peripheral reset (see the SPIiRST bits in the RCC_APBiRSTR registers).

Standard disable procedure is based on pulling BSY status together with FTLVL[1:0] to check if a transmission session is fully completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When NSS signal is managed by software and master has to provide proper end of NSS pulse for slave, or
- When transactions’ streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

The correct disable procedure is (except when receive only mode is used):

1. Wait until FTLVL[1:0] = 00 (no more data to transmit).
2. Wait until BSY=0 (the last data frame is processed).
3. Disable the SPI (SPE=0).
4. Read data until FRLVL[1:0] = 00 (read all the received data).

The correct disable procedure for certain receive only modes is:

1. Interrupt the receive flow by disabling SPI (SPE=0) in the specific time window while the last data frame is ongoing.
2. Wait until BSY=0 (the last data frame is processed).
3. Read data until FRLVL[1:0] = 00 (read all the received data).

Note: If packing mode is used and an odd number of data frames with a format less than or equal to 8 bits (fitting into one byte) has to be received, FRXTH must be set when FRLVL[1:0] = 01, in order to generate the RXNE event to read the last odd data frame and to keep good FIFO pointer alignment.

**Data packing**

When the data frame size fits into one byte (less than or equal to 8 bits), data packing is used automatically when any read or write 16-bit access is performed on the SPIx_DR register. The double data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other half stored in the MSB. **Figure 365** provides an example of data packing mode sequence handling. Two data frames are sent after the single 16-bit access the SPIx_DR register of the transmitter. This sequence can generate just one RXNE event in the receiver if the RXFIFO threshold is set to 16 bits (FRXTH=0). The receiver then has to access both data frames by a single 16-bit read of SPIx_DR as a response to this single RXNE event. The
Rx FIFO threshold setting and the following read access must be always kept aligned at the receiver side, as data can be lost if it is not in line.

A specific problem appears if an odd number of such “fit into one byte” data frames must be handled. On the transmitter side, writing the last data frame of any odd sequence with an 8-bit access to SPIx_DR is enough. The receiver has to change the Rx_FIFO threshold level for the last data frame received in the odd sequence of frames in order to generate the RXNE event.

**Figure 365. Packing data in FIFO for transmission and reception**

Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXE or RXNE enable bit in the SPIx_CR2 register is set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPIx_DR register.
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPIx_DR register.

See **Figure 366** through **Figure 369**.

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received is not read. When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or entering the Stop mode. The software must first wait until FTLVL[1:0]=00 and then until BSY=0.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:
1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI_CR2 register, if DMA Rx is used.
2. Enable DMA streams for Tx and Rx in DMA registers, if the streams are used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CR2 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order:
1. Disable DMA streams for Tx and Rx in the DMA registers, if the streams are used.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI_CR2 register, if DMA Tx and/or DMA Rx are used.

**Packing with DMA**

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPIx_CR2 register) packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel. If the DMA channel PSIZE value is equal to 16-bit and SPI data size is less than or equal to 8-bit, then packing mode is enabled. The DMA then automatically manages the write operations to the SPIx_DR register.

If data packing mode is used and the number of data to transfer is not a multiple of two, the LDMA_TX/LDMA_RX bits must be set. The SPI then considers only one data for the transmission or reception to serve the last DMA transfer (for more details refer to Data packing on page 1106.)
Communication diagrams

Some typical timing schemes are explained in this section. These schemes are valid no matter if the SPI events are handled by polling, interrupts or DMA. For simplicity, the LSBFIRST=0, CPOL=0 and CPHA=1 setting is used as a common assumption here. No complete configuration of DMA streams is provided.

The following numbered notes are common for Figure 366 on page 1110 through Figure 369 on page 1113.

1. The slave starts to control MISO line as NSS is active and SPI is enabled, and is disconnected from the line when one of them is released. Sufficient time must be provided for the slave to prepare data dedicated to the master in advance before its transaction starts.
   At the master, the SPI peripheral takes control at MOSI and SCK signals (occasionally at NSS signal as well) only if SPI is enabled. If SPI is disabled the SPI peripheral is disconnected from GPIO logic, so the levels at these lines depends on GPIO setting exclusively.

2. At the master, BSY stays active between frames if the communication (clock signal) is continuous. At the slave, BSY signal always goes down for at least one clock cycle between data frames.

3. The TXE signal is cleared only if TXFIFO is full.

4. The DMA arbitration process starts just after the TXDMAEN bit is set. The TXE interrupt is generated just after the TXEIE is set. As the TXE signal is at an active level, data transfers to TxFIFO start, until TxFIFO becomes full or the DMA transfer completes.

5. If all the data to be sent can fit into TxFIFO, the DMA Tx TCIF flag can be raised even before communication on the SPI bus starts. This flag always rises before the SPI transaction is completed.

6. The CRC value for a package is calculated continuously frame by frame in the SPIx_TXCRCR and SPIx_RXCRCR registers. The CRC information is processed after the entire data package has completed, either automatically by DMA (Tx channel must be set to the number of data frames to be processed) or by SW (the user must handle CRCNEXT bit during the last data frame processing).
   While the CRC value calculated in SPIx_TXCRCR is simply sent out by transmitter, received CRC information is loaded into RxFIFO and then compared with the SPIx_RXCRCR register content (CRC error flag can be raised here if any difference). This is why the user must take care to flush this information from the FIFO, either by software reading out all the stored content of RxFIFO, or by DMA when the proper number of data frames is preset for Rx channel (number of data frames + number of CRC frames) (see the settings at the example assumption).

7. In data packed mode, TxE and RxNE events are paired and each read/write access to the FIFO is 16 bits wide until the number of data frames are even. If the TxFIFO is ¾ full FTLVL status stays at FIFO full level. That is why the last odd data frame cannot be stored before the TxFIFO becomes ½ full. This frame is stored into TxFIFO with an 8-bit access either by software or automatically by DMA when LDMA_TX control is set.

8. To receive the last odd data frame in packed mode, the Rx threshold must be changed to 8-bit when the last data frame is processed, either by software setting FRXTH=1 or automatically by a DMA internal signal when LDMA_RX is set.
Assumptions for master full-duplex communication example:

- Data size > 8 bit

If DMA is used:
- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also: Communication diagrams on page 1109 for details about common assumptions and notes.
Assumptions for slave full-duplex communication example:

- Data size > 8 bit

If DMA is used:
- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also: Communication diagrams on page 1109 for details about common assumptions and notes.
Assumptions for master full-duplex communication with CRC example:

- Data size = 16 bit
- CRC enabled

If DMA is used:
- Number of Tx frames transacted by DMA is set to 2
- Number of Rx frames transacted by DMA is set to 3

See also: Communication diagrams on page 1109 for details about common assumptions and notes.
Figure 369. Master full-duplex communication in packed mode

Assumptions for master full-duplex communication in packed mode example:

- Data size = 5 bit
- Read/write FIFO is performed mostly by 16-bit access
- FRXTH=0

If DMA is used:

- Number of Tx frames to be transacted by DMA is set to 3
- Number of Rx frames to be transacted by DMA is set to 3
- PSIZE for both Tx and Rx DMA channel is set to 16-bit
- LDMA_TX=1 and LDMA_RX=1

See also: Communication diagrams on page 1109 for details about common assumptions and notes.
34.5.10 SPI status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TXE)

The TXE flag is set when transmission TXFIFO has enough space to store data to send. TXE flag is linked to the TXFIFO level. The flag goes high and stays high until the TXFIFO level is lower or equal to 1/2 of the FIFO depth. An interrupt can be generated if the TXEIE bit in the SPIx_CR2 register is set. The bit is cleared automatically when the TXFIFO level becomes greater than 1/2.

Rx buffer not empty (RXNE)

The RXNE flag is set depending on the FRXTH bit value in the SPIx_CR2 register:
- If FRXTH is set, RXNE goes high and stays high until the RXFIFO level is greater or equal to 1/4 (8-bit).
- If FRXTH is cleared, RXNE goes high and stays high until the RXFIFO level is greater than or equal to 1/2 (16-bit).

An interrupt can be generated if the RXNEIE bit in the SPIx_CR2 register is set.
The RXNE is cleared by hardware automatically when the above conditions are no longer true.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect).

When BSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy).
The BSY flag can be used in certain modes to detect the end of a transfer so that the software can disable the SPI or its peripheral clock before entering a low-power mode which does not provide a clock for the peripheral. This avoids corrupting the last transfer.
The BSY flag is also useful for preventing write collisions in a multimaster system.
The BSY flag is cleared under any one of the following conditions:
- When the SPI is correctly disabled
- When a fault is detected in Master mode (MODF bit set to 1)
- In Master mode, when it finishes a data transmission and no new data is ready to be sent
- In Slave mode, when the BSY flag is set to '0' for at least one SPI clock cycle between each data transfer.

Note: When the next transmission can be handled immediately by the master (e.g. if the master is in Receive-only mode or its Transmit FIFO is not empty), communication is continuous and the BSY flag remains set to '1' between transfers on the master side. Although this is not the case with a slave, it is recommended to use always the TXE and RXNE flags (instead of the BSY flags) to handle data transmission or reception operations.
34.5.11 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the ERRIE bit.

**Overrun flag (OVR)**

An overrun condition occurs when data is received by a master or slave and the RXFIFO has not enough space to store this received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RXFIFO) or when space for data storage is limited e.g. the RXFIFO is not available when CRC is enabled in receive only mode so in this case the reception buffer is limited into a single data frame buffer (see Section 34.5.14: CRC calculation).

When an overrun condition occurs, the newly received value does not overwrite the previous one in the RXFIFO. The newly received value is discarded and all data transmitted subsequently is lost. Clearing the OVR bit is done by a read access to the SPI_DR register followed by a read access to the SPI_SR register.

**Mode fault (MODF)**

Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SSI bit in NSS software mode) pulled low. This automatically sets the MODF bit. Master mode fault affects the SPI interface in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPIx_SR register while the MODF bit is set.
2. Then write to the SPIx_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence. As a security, hardware does not allow the SPE and MSTR bits to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multimaster conflict.

**CRC error (CRCERR)**

This flag is used to verify the validity of the value received when the CRCEN bit in the SPIx_CR1 register is set. The CRCERR flag in the SPIx_SR register is set if the value received in the shift register does not match the receiver SPIx_RXCRCR value. The flag is cleared by the software.

**TI mode frame format error (FRE)**

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the FRE flag is set in the SPIx_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of two data bytes.
The FRE flag is cleared when SPIx_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

### 34.5.12 NSS pulse mode

This mode is activated by the NSSP bit in the SPIx_CR2 register and it takes effect only if the SPI interface is configured as Motorola SPI master (FRF=0) with capture on the first edge (SPIx_CR1 CPHA = 0, CPOL setting is ignored). When activated, an NSS pulse is generated between two consecutive data frame transfers when NSS stays at high level for the duration of one clock period at least. This mode allows the slave to latch data. NSSP pulse mode is designed for applications with a single master-slave pair.

Figure 370 illustrates NSS pin management when NSSP pulse mode is enabled.

**Figure 370. NSSP pulse generation in Motorola SPI master mode**

![NSS pulse generation in Motorola SPI master mode](image1)

**Note:** Similar behavior is encountered when CPOL = 0. In this case the sampling edge is the rising edge of SCK, and NSS assertion and deassertion refer to this sampling edge.

### 34.5.13 TI mode

**TI protocol in master mode**

The SPI interface is compatible with the TI protocol. The FRF bit of the SPIx_CR2 register can be used to configure the SPI to be compliant with this protocol.

The clock polarity and phase are forced to conform to the TI protocol requirements whatever the values set in the SPIx_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPIx_CR1 and SPIx_CR2 registers (SSM, SSI, SSOE) impossible in this case.

In slave mode, the SPI baud rate prescaler is used to control the moment when the MISO pin state changes to HiZ when the current transaction finishes (see Figure 371). Any baud rate can be used, making it possible to determine this moment with optimal flexibility. However, the baud rate is generally set to the external master clock baud rate. The delay for the MISO signal to become HiZ (trelease) depends on internal resynchronization and on the
baud rate value set in through the BR[2:0] bits in the SPIx_CR1 register. It is given by the formula:

\[
\frac{t_{\text{baud rate}}}{2} + 4 \times t_{\text{pclk}} < t_{\text{release}} < \frac{t_{\text{baud rate}}}{2} + 6 \times t_{\text{pclk}}
\]

If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

If the data size is equal to 4-bits or 5-bits, the master in full-duplex mode or transmit-only mode uses a protocol with one more dummy data bit added after LSB. TI NSS pulse is generated above this dummy bit clock cycle instead of the LSB in each period.

This feature is not available for Motorola SPI communications (FRF bit set to 0).

*Figure 371: TI mode transfer* shows the SPI communication waveforms when TI mode is selected.

---

### 34.5.14 CRC calculation

Two separate CRC calculators are implemented in order to check the reliability of transmitted and received data. The SPI offers CRC8 or CRC16 calculation independently of the frame data length, which can be fixed to 8-bit or 16-bit. For all the other data frame lengths, no CRC is available.

**CRC principle**

CRC calculation is enabled by setting the CRCEN bit in the SPIx_CR1 register before the SPI is enabled (SPE = 1). The CRC value is calculated using an odd programmable polynomial on each bit. The calculation is processed on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx_CR1 register. The calculated CRC value is checked automatically at the end of the data block as well as for transfer managed by CPU or by the DMA. When a mismatch is detected between the CRC calculated internally on the received data and the CRC sent by the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC calculation depends on the SPI configuration and the chosen transfer management.
Note: The polynomial value should only be odd. No even values are supported.

**CRC transfer managed by CPU**

Communication starts and continues normally until the last data frame has to be sent or received in the SPIx_DR register. Then CRCNEXT bit has to be set in the SPIx_CR1 register to indicate that the CRC frame transaction will follow after the transaction of the currently processed data frame. The CRCNEXT bit must be set before the end of the last data frame transaction. CRC calculation is frozen during CRC transaction.

The received CRC is stored in the RXFIFO like a data byte or word. That is why in CRC mode only, the reception buffer has to be considered as a single 16-bit buffer used to receive only one data frame at a time.

A CRC-format transaction usually takes one more data frame to communicate at the end of data sequence. However, when setting an 8-bit data frame checked by 16-bit CRC, two more frames are necessary to send the complete CRC.

When the last CRC data is received, an automatic check is performed comparing the received value and the value in the SPIx_RXCRC register. Software has to check the CRCERR flag in the SPIx_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing '0' to it.

After the CRC reception, the CRC value is stored in the RXFIFO and must be read in the SPIx_DR register in order to clear the RXNE flag.

**CRC transfer managed by DMA**

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication is automatic (with the exception of reading CRC data in receive only mode). The CRCNEXT bit does not have to be handled by the software. The counter for the SPI transmission DMA channel has to be set to the number of data frames to transmit excluding the CRC frame. On the receiver side, the received CRC value is handled automatically by DMA at the end of the transaction but user must take care to flush out received CRC information from RXFIFO as it is always loaded into it. In full-duplex mode, the counter of the reception DMA channel can be set to the number of data frames to receive including the CRC, which means, for example, in the specific case of an 8-bit data frame checked by 16-bit CRC:

\[
\text{DMA}_\text{RX} = \text{Numb}_\text{of}_\text{data} + 2
\]

In receive only mode, the DMA reception channel counter should contain only the amount of data transferred, excluding the CRC calculation. Then based on the complete transfer from DMA, all the CRC values must be read back by software from FIFO as it works as a single buffer in this mode.

At the end of the data and CRC transfers, the CRCERR flag in the SPIx_SR register is set if corruption occurred during the transfer.

If packing mode is used, the LDMA_RX bit needs managing if the number of data is odd.

**Resetting the SPIx_TXCRC and SPIx_RXCRC values**

The SPIx_TXCRC and SPIx_RXCRC values are cleared automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode (not available in receive-only mode) in order to transfer data without any interruption, (several data blocks covered by intermediate CRC checking phases).
If the SPI is disabled during a communication the following sequence must be followed:
1. Disable the SPI
2. Clear the CRCEN bit
3. Enable the CRCEN bit
4. Enable the SPI

Note: When the SPI interface is configured as a slave, the NSS internal signal needs to be kept low during transaction of the CRC phase once the CRCNEXT signal is released. That is why the CRC calculation cannot be used at NSS Pulse mode when NSS hardware mode should be applied at slave normally.

At TI mode, despite the fact that clock phase and clock polarity setting is fixed and independent on SPIx_CR1 register, the corresponding setting CPOL=0 CPHA=1 has to be kept at the SPIx_CR1 register anyway if CRC is applied. In addition, the CRC calculation has to be reset between sessions by SPI disable sequence with re-enable the CRCEN bit described above at both master and slave side, else CRC calculation can be corrupted at this specific mode.

34.6 SPI interrupts

During SPI communication an interrupt can be generated by the following events:
- Transmit TXFIFO ready to be loaded
- Data received in Receive RXFIFO
- Master mode fault
- Overrun error
- TI frame format error
- CRC protocol error

Interrupts can be enabled and disabled separately.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit TXFIFO ready to be loaded</td>
<td>TXE</td>
<td>TXEIE</td>
</tr>
<tr>
<td>Data received in RXFIFO</td>
<td>RXNE</td>
<td>RXNEIE</td>
</tr>
<tr>
<td>Master Mode fault event</td>
<td>MODF</td>
<td></td>
</tr>
<tr>
<td>Overrun error</td>
<td>OVR</td>
<td>ERRIE</td>
</tr>
<tr>
<td>TI frame format error</td>
<td>FRE</td>
<td></td>
</tr>
<tr>
<td>CRC protocol error</td>
<td>CRCERR</td>
<td></td>
</tr>
</tbody>
</table>
34.7 I2S functional description

34.7.1 I2S general description

The block diagram of the I2S is shown in Figure 372.

Figure 372. I2S block diagram

1. MCK is mapped on the MISO pin.

The SPI can function as an audio I2S interface when the I2S capability is enabled (by setting the I2SMOD bit in the SPIx_I2SCFGR register). This interface mainly uses the same pins, flags and interrupts as the SPI.
The I2S shares three common pins with the SPI:

- **SD**: Serial Data (mapped on the MOSI pin) to transmit or receive the two time-multiplexed data channels (in half-duplex mode only).
- **WS**: Word Select (mapped on the NSS pin) is the data control signal output in master mode and input in slave mode.
- **CK**: Serial Clock (mapped on the SCK pin) is the serial clock output in master mode and serial clock input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- **MCK**: Master Clock (mapped separately) is used, when the I2S is configured in master mode (and when the MCKOE bit in the SPIx_I2SPR register is set), to output this additional clock generated at a preconfigured frequency rate equal to $256 \times f_S$, where $f_S$ is the audio sampling frequency.

The I2S uses its own clock generator to produce the communication clock when it is set in master mode. This clock generator is also the source of the master clock output. Two additional registers are available in I2S mode. One is linked to the clock generator configuration SPIx_I2SPR and the other one is a generic I2S configuration register SPIx_I2SCFGR (audio standard, slave/master mode, data format, packet frame, clock polarity, etc.).

The SPIx_CR1 register and all CRC registers are not used in the I2S mode. Likewise, the SSOE bit in the SPIx_CR2 register and the MODF and CRCERR bits in the SPIx_SR are not used.

The I2S uses the same SPI register for data transfer (SPIx_DR) in 16-bit wide mode.

### 34.7.2 Supported audio protocols

The three-line bus has to handle only audio data generally time-multiplexed on two channels: the right channel and the left channel. However there is only one 16-bit register for transmission or reception. So, it is up to the software to write into the data register the appropriate value corresponding to each channel side, or to read the data from the data register and to identify the corresponding channel by checking the CHSIDE bit in the SPIx_SR register. Channel left is always sent first followed by the channel right (CHSIDE has no meaning for the PCM protocol).

Four data and packet frames are available. Data may be sent with a format of:

- 16-bit data packed in a 16-bit frame
- 16-bit data packed in a 32-bit frame
- 24-bit data packed in a 32-bit frame
- 32-bit data packed in a 32-bit frame

When using 16-bit data extended on 32-bit packet, the first 16 bits (MSB) are the significant bits, the 16-bit LSB is forced to 0 without any need for software action or DMA request (only one read/write operation).

The 24-bit and 32-bit data frames need two CPU read or write operations to/from the SPIx_DR register or two DMA operations if the DMA is preferred for the application. For 24-bit data frame specifically, the 8 non-significant bits are extended to 32 bits with 0-bits (by hardware).
For all data formats and communication standards, the most significant bit is always sent first (MSB first).

The I²S interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPIx_I2SCFGR register.

**I²S Philips standard**

For this standard, the WS signal is used to indicate which channel is being transmitted. It is activated one CK clock cycle before the first bit (MSB) is available.

**Figure 373. I²S Philips protocol waveforms (16/32-bit full accuracy)**

![I²S Philips protocol waveforms](image1)

Data are latched on the falling edge of CK (for the transmitter) and are read on the rising edge (for the receiver). The WS signal is also latched on the falling edge of CK.

**Figure 374. I²S Philips standard waveforms (24-bit frame)**

![I²S Philips standard waveforms](image2)

This mode needs two write or read operations to/from the SPIx_DR register.

- In transmission mode:
  - If 0x8EAA33 has to be sent (24-bit):
• In reception mode:
  If data 0x8EAA33 is received:

  **Figure 376. Receiving 0x8EAA33**

  First read to Data register  
  Second read to Data register

  0x8EAA 0x33XX
  Only the 8 MSB are sent to compare the 24 bits
  8 LSBs have no meaning and can be anything

When 16-bit data frame extended to 32-bit channel frame is selected during the I2S configuration phase, only one access to the SPIx_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

If the data to transmit or the received data are 0x76A3 (0x76A30000 extended to 32-bit), the operation shown in **Figure 378** is required.

**Figure 378. Example of 16-bit data frame extended to 32-bit channel frame**

  Only one access to SPIx_DR

  0x76A3
For transmission, each time an MSB is written to SPIx_DR, the TXE flag is set and its interrupt, if allowed, is generated to load the SPIx_DR register with the new value to send. This takes place even if 0x0000 have not yet been sent because it is done by hardware.

For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

In this way, more time is provided between two write or read operations, which prevents underrun or overrun conditions (depending on the direction of the data transfer).

**MSB justified standard**

For this standard, the WS signal is generated at the same time as the first data bit, which is the MSBit.

**Figure 379. MSB Justified 16-bit or 32-bit full-accuracy length**

![Figure 379. MSB Justified 16-bit or 32-bit full-accuracy length](image)

Data are latched on the falling edge of CK (for transmitter) and are read on the rising edge (for the receiver).

**Figure 380. MSB justified 24-bit frame length**

![Figure 380. MSB justified 24-bit frame length](image)
Figure 381. MSB justified 16-bit extended to 32-bit packet frame

LSB justified standard
This standard is similar to the MSB justified standard (no difference for the 16-bit and 32-bit full-accuracy frame formats).
The sampling of the input and output signals is the same as for the I²S Philips standard.

Figure 382. LSB justified 16-bit or 32-bit full-accuracy

Figure 383. LSB justified 24-bit frame length

• In transmission mode:
If data 0x3478AE have to be transmitted, two write operations to the SPIx_DR register are required by software or by DMA. The operations are shown below.
• In reception mode:
  If data 0x3478AE are received, two successive read operations from the SPIx_DR register are required on each RXNE event.

**Figure 385. Operations required to receive 0x3478AE**

- First read from Data register conditioned by RXNE=1
  - 0xXX34
  - Only the 8 LSB of the half-word are significant. A field of 0x00 is forced instead of the 8 MSBs.

- Second read from Data register conditioned by RXNE=1
  - 0x78AE

When 16-bit data frame extended to 32-bit channel frame is selected during the I2S configuration phase, Only one access to the SPIx_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format. In this case it corresponds to the half-word MSB.

If the data to transmit or the received data are 0x76A3 (0x0000 76A3 extended to 32-bit), the operation shown in **Figure 387** is required.
In transmission mode, when a TXE event occurs, the application has to write the data to be transmitted (in this case 0x76A3). The 0x000 field is transmitted first (extension on 32-bit). The TXE flag is set again as soon as the effective data (0x76A3) is sent on SD.

In reception mode, RXNE is asserted as soon as the significant half-word is received (and not the 0x0000 field).

In this way, more time is provided between two write or read operations to prevent underrun or overrun conditions.

**PCM standard**

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and configurable using the PCMSYNC bit in SPIx_I2SCFGR register.

In PCM mode, the output signals (WS, SD) are sampled on the rising edge of CK signal. The input signals (WS, SD) are captured on the falling edge of CK.

Note that CK and WS are configured as output in MASTER mode.

For long frame synchronization, the WS signal assertion time is fixed to 13 bits in master mode.

For short frame synchronization, the WS synchronization signal is only one cycle long.
Note: For both modes (master and slave) and for both synchronizations (short and long), the number of bits between two consecutive pieces of data (and so two synchronization signals) needs to be specified (DATLEN and CHLEN bits in the SPIx_I2SCFGGR register) even in slave mode.

34.7.3 Start-up description

The Figure 390 shows how the serial interface is handled in MASTER mode, when the SPI/I2S is enabled (via I2SE bit). It shows as well the effect of CKPOL on the generated signals.
In slave mode, the way the frame synchronization is detected, depends on the value of ASTRTEN bit.

If ASTRTEN = 0, when the audio interface is enabled (I2SE = 1), then the hardware waits for the appropriate transition on the incoming WS signal, using the CK signal.
The appropriate transition is a falling edge on WS signal when I²S Philips Standard is used, or a rising edge for other standards. The falling edge is detected by sampling first WS to 1 and then to 0, and vice-versa for the rising edge detection.

If ASRTEN = 1, the user has to enable the audio interface before the WS becomes active. This means that the I2SE bit must be set to 1 when WS = 1 for I²S Philips standard, or when WS = 0 for other standards.

### 34.7.4 Clock generator

The I²S bit rate determines the data flow on the I²S data line and the I²S clock signal frequency.

I²S bit rate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio, left and right channel, the I²S bit rate is calculated as follows:

I²S bit rate = 16 × 2 × f_s

It will be: I²S bit rate = 32 × 2 × f_s if the packet length is 32-bit wide.

**Figure 391. Audio sampling frequency definition**

When the master mode is configured, a specific action needs to be taken to properly program the linear divider in order to communicate with the desired audio frequency.

**Figure 392. I²S clock generator architecture**
Figure 392 presents the communication clock architecture. The I2SxCLK clock is provided by the reset and clock controller (RCC) of the product. The I2SxCLK clock can be asynchronous with respect to the SPI/I2S APB clock.

**Warning:** In addition, it is mandatory to keep the I2SxCLK frequency higher or equal to the APB clock used by the SPI/I2S block. If this condition is not respected, the SPI/I2S will not work properly.

The audio sampling frequency may be 192 kHz, 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz or 8 kHz (or any other value within this range).

In order to reach the desired frequency, the linear divider needs to be programmed according to the formulas below:

**For I²S modes:**

When the master clock is generated (MCKOE in the SPIx_I2SPR register is set):

\[
F_s = \frac{F_{I2SxCLK}}{256 \times ((2 \times I2SDIV) + ODD)}
\]

When the master clock is disabled (MCKOE bit cleared):

\[
F_s = \frac{F_{I2SxCLK}}{32 \times (CHLEN + 1) \times ((2 \times I2SDIV) + ODD)}
\]

CHLEN = 0 when the channel frame is 16-bit wide and,
CHLEN = 1 when the channel frame is 32-bit wide.

**For PCM modes:**

When the master clock is generated (MCKOE in the SPIx_I2SPR register is set):

\[
F_s = \frac{F_{I2SxCLK}}{128 \times ((2 \times I2SDIV) + ODD)}
\]

When the master clock is disabled (MCKOE bit cleared):

\[
F_s = \frac{F_{I2SxCLK}}{16 \times (CHLEN + 1) \times ((2 \times I2SDIV) + ODD)}
\]

CHLEN = 0 when the channel frame is 16-bit wide and,
CHLEN = 1 when the channel frame is 32-bit wide.

Where \(F_s\) is the audio sampling frequency, and \(F_{I2SxCLK}\) is the frequency of the kernel clock provided to the SPI/I2S block.
Note: Note that I2SDIV must be strictly higher than 1.

Table 183 provides example precision values for different clock configurations.

Note: Other configurations are possible that allow optimum clock precision.

<table>
<thead>
<tr>
<th>SYCLK (MHz)</th>
<th>Data length</th>
<th>I2SDIV</th>
<th>I2SODD</th>
<th>MCLK</th>
<th>Target fs (Hz)</th>
<th>Real fs (kHz)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td>No</td>
<td>96000</td>
<td>93750</td>
<td>2.3438%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>15</td>
<td>1</td>
<td>No</td>
<td>48000</td>
<td>48387.0968</td>
<td>0.8065%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td>No</td>
<td>48000</td>
<td>46875</td>
<td>2.3438%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>17</td>
<td>0</td>
<td>No</td>
<td>44100</td>
<td>44117.647</td>
<td>0.0400%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>8</td>
<td>1</td>
<td>No</td>
<td>44100</td>
<td>44117.647</td>
<td>0.0400%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>23</td>
<td>1</td>
<td>No</td>
<td>32000</td>
<td>31914.8936</td>
<td>0.2660%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>11</td>
<td>1</td>
<td>No</td>
<td>32000</td>
<td>32608.696</td>
<td>1.9022%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>34</td>
<td>0</td>
<td>No</td>
<td>22050</td>
<td>22058.8235</td>
<td>0.0400%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>17</td>
<td>0</td>
<td>No</td>
<td>22050</td>
<td>22058.8235</td>
<td>0.0400%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>47</td>
<td>0</td>
<td>No</td>
<td>16000</td>
<td>15957.4468</td>
<td>0.2660%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>68</td>
<td>0</td>
<td>No</td>
<td>11025</td>
<td>11029.4118</td>
<td>0.0400%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>94</td>
<td>0</td>
<td>No</td>
<td>8000</td>
<td>7978.7234</td>
<td>0.2660%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>47</td>
<td>0</td>
<td>No</td>
<td>8000</td>
<td>7978.7234</td>
<td>0.2660%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>2</td>
<td>0</td>
<td>Yes</td>
<td>48000</td>
<td>46875</td>
<td>2.3430%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>2</td>
<td>0</td>
<td>Yes</td>
<td>48000</td>
<td>46875</td>
<td>2.3430%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>2</td>
<td>0</td>
<td>Yes</td>
<td>44100</td>
<td>46875</td>
<td>6.2925%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>3</td>
<td>0</td>
<td>Yes</td>
<td>32000</td>
<td>31250</td>
<td>2.3438%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>3</td>
<td>0</td>
<td>Yes</td>
<td>32000</td>
<td>31250</td>
<td>2.3438%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td>Yes</td>
<td>22050</td>
<td>20833.333</td>
<td>5.5178%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td>Yes</td>
<td>22050</td>
<td>20833.333</td>
<td>5.5178%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>6</td>
<td>0</td>
<td>Yes</td>
<td>16000</td>
<td>15625</td>
<td>2.3438%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>6</td>
<td>0</td>
<td>Yes</td>
<td>16000</td>
<td>15625</td>
<td>2.3438%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>8</td>
<td>1</td>
<td>Yes</td>
<td>11025</td>
<td>11029.4118</td>
<td>0.0400%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>8</td>
<td>1</td>
<td>Yes</td>
<td>11025</td>
<td>11029.4118</td>
<td>0.0400%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>11</td>
<td>1</td>
<td>Yes</td>
<td>8000</td>
<td>8152.17391</td>
<td>1.9022%</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>11</td>
<td>1</td>
<td>Yes</td>
<td>8000</td>
<td>8152.17391</td>
<td>1.9022%</td>
</tr>
</tbody>
</table>
34.7.5 I2S master mode

The I2S can be configured in master mode. This means that the serial clock is generated on the CK pin as well as the Word Select signal WS. Master clock (MCK) may be output or not, controlled by the MCKOE bit in the SPIx_I2SPR register.

Procedure

1. Select the I2SDIV[7:0] bits in the SPIx_I2SPR register to define the serial clock baud rate to reach the proper audio sample frequency. The ODD bit in the SPIx_I2SPR register also has to be defined.
2. Select the CKPOL bit to define the steady level for the communication clock. Set the MCKOE bit in the SPIx_I2SPR register if the master clock MCK needs to be provided to the external DAC/ADC audio component (the I2SDIV and ODD values should be computed depending on the state of the MCK output, for more details refer to Section 34.7.4: Clock generator).
3. Set the I2SMOD bit in the SPIx_I2SCFGR register to activate the I2S functions and choose the I2S standard through the I2SSTD[1:0] and PCMSYNC bits, the data length through the DATLEN[1:0] bits and the number of bits per channel by configuring the CHLEN bit. Select also the I2S master mode and direction (Transmitter or Receiver) through the I2SCFG[1:0] bits in the SPIx_I2SCFGR register.
4. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CR2 register.
5. The I2SE bit in SPIx_I2SCFGR register must be set.

WS and CK are configured in output mode. MCK is also an output, if the MCKOE bit in SPIx_I2SPR is set.

Transmission sequence

The transmission sequence begins when a half-word is written into the Tx buffer.

Let's assume the first data written into the Tx buffer corresponds to the left channel data. When data are transferred from the Tx buffer to the shift register, TXE is set and data corresponding to the right channel have to be written into the Tx buffer. The CHSIDE flag indicates which channel is to be transmitted. It has a meaning when the TXE flag is set because the CHSIDE flag is updated when TXE goes high.

A full frame has to be considered as a left channel data transmission followed by a right channel data transmission. It is not possible to have a partial frame where only the left channel is sent.

The data half-word is parallel loaded into the 16-bit shift register during the first bit transmission, and then shifted out, serially, to the MOSI/SD pin, MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx_CR2 register is set.

For more details about the write operations depending on the I2S standard mode selected, refer to Section 34.7.2: Supported audio protocols.

To ensure a continuous audio data transmission, it is mandatory to write the SPIx_DR register with the next data to transmit before the end of the current transmission.

1. This table gives only example values for different clock configurations. Other configurations allowing optimum clock precision are possible.
To switch off the I2S, by clearing I2SE, it is mandatory to wait for TXE = 1 and BSY = 0.

**Reception sequence**

The operating mode is the same as for transmission mode except for the point 3 (refer to the procedure described in Section 34.7.5: I2S master mode), where the configuration should set the master reception mode through the I2SCFG[1:0] bits.

Whatever the data or channel length, the audio data are received by 16-bit packets. This means that each time the Rx buffer is full, the RXNE flag is set and an interrupt is generated if the RXNEIE bit is set in SPIx_CR2 register. Depending on the data and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the Rx buffer.

Clearing the RXNE bit is performed by reading the SPIx_DR register.

CHSIDE is updated after each reception. It is sensitive to the WS signal generated by the I2S cell.

For more details about the read operations depending on the I2S standard mode selected, refer to Section 34.7.2: Supported audio protocols.

If data are received while the previously received data have not been read yet, an overrun is generated and the OVR flag is set. If the ERRIE bit is set in the SPIx_CR2 register, an interrupt is generated to indicate the error.

To switch off the I2S, specific actions are required to ensure that the I2S completes the transfer cycle properly without initiating a new data transfer. The sequence depends on the configuration of the data and channel lengths, and on the audio protocol mode selected. In the case of:

- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) using the LSB justified mode (I2SSTD = 10)
  a) Wait for the second to last RXNE = 1 (n – 1)
  b) Then wait 17 I2S clock cycles (using a software loop)
  c) Disable the I2S (I2SE = 0)
- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) in MSB justified, I2S or PCM modes (I2SSTD = 00, I2SSTD = 01 or I2SSTD = 11, respectively)
  a) Wait for the last RXNE
  b) Then wait 1 I2S clock cycle (using a software loop)
  c) Disable the I2S (I2SE = 0)
- For all other combinations of DATLEN and CHLEN, whatever the audio mode selected through the I2SSTD bits, carry out the following sequence to switch off the I2S:
  a) Wait for the second to last RXNE = 1 (n – 1)
  b) Then wait one I2S clock cycle (using a software loop)
  c) Disable the I2S (I2SE = 0)

**Note:** The BSY flag is kept low during transfers.

### 34.7.6 I2S slave mode

For the slave configuration, the I2S can be configured in transmission or reception mode. The operating mode is following mainly the same rules as described for the I2S master.
configuration. In slave mode, there is no clock to be generated by the I2S interface. The clock and WS signals are input from the external master connected to the I2S interface. There is then no need, for the user, to configure the clock.

The configuration steps to follow are listed below:

1. Set the I2SMOD bit in the SPIx_I2SCFGR register to select I²S mode and choose the I²S standard through the I2SSTD[1:0] bits, the data length through the DATLEN[1:0] bits and the number of bits per channel for the frame configuring the CHLEN bit. Select also the mode (transmission or reception) for the slave through the I2SCFG[1:0] bits in SPIx_I2SCFGR register.

2. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CR2 register.

3. The I2SE bit in SPIx_I2SCFGR register must be set.

Transmission sequence

The transmission sequence begins when the external master device sends the clock and when the NSS_WS signal requests the transfer of data. The slave has to be enabled before the external master starts the communication. The I2S data register has to be loaded before the master initiates the communication.

For the I2S, MSB justified and LSB justified modes, the first data item to be written into the data register corresponds to the data for the left channel. When the communication starts, the data are transferred from the Tx buffer to the shift register. The TXE flag is then set in order to request the right channel data to be written into the I2S data register.

The CHSIDE flag indicates which channel is to be transmitted. Compared to the master transmission mode, in slave mode, CHSIDE is sensitive to the WS signal coming from the external master. This means that the slave needs to be ready to transmit the first data before the clock is generated by the master. WS assertion corresponds to left channel transmitted first.

Note: The I2SE has to be written at least two PCLK cycles before the first clock of the master comes on the CK line.

The data half-word is parallel-loaded into the 16-bit shift register (from the internal bus) during the first bit transmission, and then shifted out serially to the MOSI/SD pin MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx_CR2 register is set.

Note that the TXE flag should be checked to be at 1 before attempting to write the Tx buffer.

For more details about the write operations depending on the I²S standard mode selected, refer to Section 34.7.2: Supported audio protocols.

To secure a continuous audio data transmission, it is mandatory to write the SPIx_DR register with the next data to transmit before the end of the current transmission. An underrun flag is set and an interrupt may be generated if the data are not written into the SPIx_DR register before the first clock edge of the next data communication. This indicates to the software that the transferred data are wrong. If the ERRIE bit is set into the SPIx_CR2 register, an interrupt is generated when the UDR flag in the SPIx_SR register goes high. In this case, it is mandatory to switch off the I2S and to restart a data transfer starting from the left channel.

To switch off the I2S, by clearing the I2SE bit, it is mandatory to wait for TXE = 1 and BSY = 0.
Reception sequence

The operating mode is the same as for the transmission mode except for the point 1 (refer to the procedure described in Section 34.7.6: I2S slave mode), where the configuration should set the master reception mode using the I2SCFG[1:0] bits in the SPIx_I2SCFGR register. Whatever the data length or the channel length, the audio data are received by 16-bit packets. This means that each time the RX buffer is full, the RXNE flag in the SPIx SR register is set and an interrupt is generated if the RXNEIE bit is set in the SPIx_CR2 register. Depending on the data length and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the RX buffer. The CHSIDE flag is updated each time data are received to be read from the SPIx_DR register. It is sensitive to the external WS line managed by the external master component. Clearing the RXNE bit is performed by reading the SPIx_DR register. For more details about the read operations depending the I2S standard mode selected, refer to Section 34.7.2: Supported audio protocols.

If data are received while the preceding received data have not yet been read, an overrun is generated and the OVR flag is set. If the bit ERRIE is set in the SPIx_CR2 register, an interrupt is generated to indicate the error.

To switch off the I2S in reception mode, I2SE has to be cleared immediately after receiving the last RXNE = 1.

Note: The external master components should have the capability of sending/receiving data in 16-bit or 32-bit packets via an audio channel.

34.7.7 I2S status flags

Three status flags are provided for the application to fully monitor the state of the I2S bus.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect). It indicates the state of the communication layer of the I2S.

When BSY is set, it indicates that the I2S is busy communicating. There is one exception in master receive mode (I2SCFG = 11) where the BSY flag is kept low during reception.

The BSY flag is useful to detect the end of a transfer if the software needs to disable the I2S. This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is set when a transfer starts, except when the I2S is in master receiver mode.

The BSY flag is cleared:

- When a transfer completes (except in master transmit mode, in which the communication is supposed to be continuous)
- When the I2S is disabled

When communication is continuous:

- In master transmit mode, the BSY flag is kept high during all the transfers
- In slave mode, the BSY flag goes low for one I2S clock cycle between each transfer

Note: Do not use the BSY flag to handle each data transmission or reception. It is better to use the TXE and RXNE flags instead.
Tx buffer empty flag (TXE)
When set, this flag indicates that the Tx buffer is empty and the next data to be transmitted can then be loaded into it. The TXE flag is reset when the Tx buffer already contains data to be transmitted. It is also reset when the I2S is disabled (I2SE bit is reset).

RX buffer not empty (RXNE)
When set, this flag indicates that there are valid received data in the RX Buffer. It is reset when SPIx_DR register is read.

Channel Side flag (CHSIDE)
In transmission mode, this flag is refreshed when TXE goes high. It indicates the channel side to which the data to transfer on SD has to belong. In case of an underrun error event in slave transmission mode, this flag is not reliable and I2S needs to be switched off and switched on before resuming the communication.

In reception mode, this flag is refreshed when data are received into SPIx_DR. It indicates from which channel side data have been received. Note that in case of error (like OVR) this flag becomes meaningless and the I2S should be reset by disabling and then enabling it (with configuration if it needs changing).

This flag has no meaning in the PCM standard (for both Short and Long frame modes).

When the OVR or UDR flag in the SPIx_SR is set and the ERRIE bit in SPIx_CR2 is also set, an interrupt is generated. This interrupt can be cleared by reading the SPIx_SR status register (once the interrupt source has been cleared).

34.7.8 I2S error flags
There are three error flags for the I2S cell.

Underrun flag (UDR)
In slave transmission mode this flag is set when the first clock for data transmission appears while the software has not yet loaded any value into SPIx_DR. It is available when the I2SMOD bit in the SPIx_I2SCFGR register is set. An interrupt may be generated if the ERRIE bit in the SPIx_CR2 register is set.
The UDR bit is cleared by a read operation on the SPIx_SR register.

Overrun flag (OVR)
This flag is set when data are received and the previous data have not yet been read from the SPIx_DR register. As a result, the incoming data are lost. An interrupt may be generated if the ERRIE bit is set in the SPIx_CR2 register.

In this case, the receive buffer contents are not updated with the newly received data from the transmitter device. A read operation to the SPIx_DR register returns the previous correctly received data. All other subsequently transmitted half-words are lost.

Clearing the OVR bit is done by a read operation on the SPIx_DR register followed by a read access to the SPIx_SR register.

Frame error flag (FRE)
This flag can be set by hardware only if the I2S is configured in Slave mode. It is set if the external master is changing the WS line while the slave is not expecting this change. If the
synchronization is lost, the following steps are required to recover from this state and 
resynchronize the external master device with the I2S slave device:
1. Disable the I2S.
2. Enable it again when the correct level is detected on the WS line (WS line is high in I²S 
mode or low for MSB- or LSB-justified or PCM modes.

Desynchronization between master and slave devices may be due to noisy environment on 
the CK communication clock or on the WS frame synchronization line. An error interrupt can 
be generated if the ERRIE bit is set. The desynchronization flag (FRE) is cleared by 
software when the status register is read.

### 34.7.9 DMA features

In I²S mode, the DMA works in exactly the same way as it does in SPI mode. There is no 
difference except that the CRC feature is not available in I²S mode since there is no data 
transfer protection system.

### 34.8 I2S interrupts

*Table 184* provides the list of I2S interrupts.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit buffer empty flag</td>
<td>TXE</td>
<td>TXEIE</td>
</tr>
<tr>
<td>Receive buffer not empty flag</td>
<td>RXNE</td>
<td>RXNEIE</td>
</tr>
<tr>
<td>Overrun error</td>
<td>OVR</td>
<td></td>
</tr>
<tr>
<td>Underrun error</td>
<td>UDR</td>
<td>ERRIE</td>
</tr>
<tr>
<td>Frame error flag</td>
<td>FRE</td>
<td></td>
</tr>
</tbody>
</table>
### 34.9 SPI and I2S registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit). SPI_DR in addition can be accessed by 8-bit access.

#### 34.9.1 SPI control register 1 (SPIx_CR1)

Address offset: 0x000

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>BIDIMODE: Bidirectional data mode enable.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit enables half-duplex communication using common single bidirectional data line.</td>
</tr>
<tr>
<td></td>
<td>Keep RXONLY bit clear when bidirectional mode is active.</td>
</tr>
<tr>
<td></td>
<td>0: 2-line unidirectional data mode selected</td>
</tr>
<tr>
<td></td>
<td>1: 1-line bidirectional data mode selected</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This bit is not used in I2S mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 14</th>
<th>BIDIOE: Output enable in bidirectional mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit combined with the BIDIMODE bit selects the direction of transfer in bidirectional mode.</td>
</tr>
<tr>
<td></td>
<td>0: Output disabled (receive-only mode)</td>
</tr>
<tr>
<td></td>
<td>1: Output enabled (transmit-only mode)</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> In master mode, the MOSI pin is used and in slave mode, the MISO pin is used.</td>
</tr>
<tr>
<td></td>
<td>This bit is not used in I2S mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 13</th>
<th>CRCEN: Hardware CRC calculation enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: CRC calculation disabled</td>
</tr>
<tr>
<td></td>
<td>1: CRC calculation enabled</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This bit should be written only when SPI is disabled (SPE = '0') for correct operation.</td>
</tr>
<tr>
<td></td>
<td>This bit is not used in I2S mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 12</th>
<th>CRCNEXT: Transmit CRC next</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: Next transmit value is from Tx buffer.</td>
</tr>
<tr>
<td></td>
<td>1: Next transmit value is from Tx CRC register.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This bit has to be written as soon as the last data is written in the SPIx_DR register.</td>
</tr>
<tr>
<td></td>
<td>This bit is not used in I2S mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 11</th>
<th>CRCL: CRC length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and cleared by software to select the CRC length.</td>
</tr>
<tr>
<td></td>
<td>0: 8-bit CRC length</td>
</tr>
<tr>
<td></td>
<td>1: 16-bit CRC length</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This bit should be written only when SPI is disabled (SPE = '0') for correct operation.</td>
</tr>
<tr>
<td></td>
<td>This bit is not used in I2S mode.</td>
</tr>
</tbody>
</table>
Bit 10  **RXONLY:** Receive only mode enabled.
This bit enables simplex communication using a single unidirectional line to receive data exclusively. Keep BIDIMODE bit clear when receive only mode is active. This bit is also useful in a multislice system in which this particular slave is not accessed, the output from the accessed slave is not corrupted.

- 0: Full-duplex (Transmit and receive)
- 1: Output disabled (Receive-only mode)

*Note:* This bit is not used in I2S mode.

Bit 9  **SSM:** Software slave management
When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit.

- 0: Software slave management disabled
- 1: Software slave management enabled

*Note:* This bit is not used in I2S mode and SPI TI mode.

Bit 8  **SSI:** Internal slave select
This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the I/O value of the NSS pin is ignored.

*Note:* This bit is not used in I2S mode and SPI TI mode.

Bit 7  **LSBFIRST:** Frame format

- 0: data is transmitted / received with the MSB first
- 1: data is transmitted / received with the LSB first

*Note:* 1. This bit should not be changed when communication is ongoing.
2. This bit is not used in I2S mode and SPI TI mode.

Bit 6  **SPE:** SPI enable

- 0: Peripheral disabled
- 1: Peripheral enabled

*Note:* When disabling the SPI, follow the procedure described in Procedure for disabling the SPI on page 1105.
This bit is not used in I2S mode.

Bits 5:3  **BR[2:0]:** Baud rate control

- 000: fPCLK/2
- 001: fPCLK/4
- 010: fPCLK/8
- 011: fPCLK/16
- 100: fPCLK/32
- 101: fPCLK/64
- 110: fPCLK/128
- 111: fPCLK/256

*Note:* These bits should not be changed when communication is ongoing.
This bit is not used in I2S mode.

Bit 2  **MSTR:** Master selection

- 0: Slave configuration
- 1: Master configuration

*Note:* This bit should not be changed when communication is ongoing.
This bit is not used in I2S mode.
34.9.2 SPI control register 2 (SPIx_CR2)

Address offset: 0x04
Reset value: 0x0700

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 14    | LDMA_TX: Last DMA transfer for transmission | Last DMA transfer for transmission | This bit is used in data packing mode, to define if the total number of data to transmit by DMA is odd or even. It has significance only if the RXDMAEN bit in the SPIx_CR2 register is set and if packing mode is used (data length <= 8-bit and write access to SPIx_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx_CR1 register). 0: Number of data to transfer is even 1: Number of data to transfer is odd  
Note: Refer to Procedure for disabling the SPI on page 1105 if the CRCEN bit is set.  
This bit is not used in I²S mode. |
| 13    | LDMA_RX: Last DMA transfer for reception | Last DMA transfer for reception | This bit is used in data packing mode, to define if the total number of data to receive by DMA is odd or even. It has significance only if the RXDMAEN bit in the SPIx_CR2 register is set and if packing mode is used (data length <= 8-bit and write access to SPIx_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx_CR1 register). 0: Number of data to transfer is even 1: Number of data to transfer is odd  
Note: Refer to Procedure for disabling the SPI on page 1105 if the CRCEN bit is set.  
This bit is not used in I²S mode. |
| 12    | FRXTH: FIFO reception threshold | FIFO reception threshold | This bit is used to set the threshold of the RXFIFO that triggers an RXNE event 0: RXNE event is generated if the FIFO level is greater than or equal to 1/2 (16-bit) 1: RXNE event is generated if the FIFO level is greater than or equal to 1/4 (8-bit)  
Note: This bit is not used in I²S mode. |
| 11    | TXEIE | TXEIE |                      |                                                                      |
| 10    | RXNEIE | RXNEIE |                      |                                                                      |
| 9     | ERRIE | ERRIE |                      |                                                                      |
| 8     | FRF | FRF |                      |                                                                      |
| 7     | NSSP | NSSP |                      |                                                                      |
| 6     | SSOE | SSOE |                      |                                                                      |
| 5     | TXDMAEN | TXDMAEN |                      |                                                                      |
| 4     | RXDMAEN | RXDMAEN |                      |                                                                      |
| 3     | 2     | 1     | 0     |                    |                                                                      |
Bits 11:8 DS[3:0]: Data size
These bits configure the data length for SPI transfers.
0000: Not used
0001: Not used
0010: Not used
0011: 4-bit
0100: 5-bit
0101: 6-bit
0110: 7-bit
0111: 8-bit
1000: 9-bit
1001: 10-bit
1010: 11-bit
1011: 12-bit
1100: 13-bit
1101: 14-bit
1110: 15-bit
1111: 16-bit
If software attempts to write one of the “Not used” values, they are forced to the value “0111” (8-bit)

*Note:* This bit is not used in I2S mode.

Bit 7 TXEIE: Tx buffer empty interrupt enable
0: TXE interrupt masked
1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.

Bit 6 RXNEIE: RX buffer not empty interrupt enable
0: RXNE interrupt masked
1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set.

Bit 5 ERRIE: Error interrupt enable
This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode, FRE at TI mode and UDR, OVR, and FRE in I2S mode).
0: Error interrupt is masked
1: Error interrupt is enabled

Bit 4 FRF: Frame format
0: SPI Motorola mode
1 SPI TI mode

*Note:* This bit must be written only when the SPI is disabled (SPE=0).
This bit is not used in I2S mode.

Bit 3 NSSP: NSS pulse management
This bit is used in master mode only, it allows the SPI to generate an NSS pulse between two consecutive data when doing continuous transfers. In the case of a single data transfer, it forces the NSS pin high level after the transfer.
It has no meaning if CPHA = ‘1’, or FRF = ‘1’.
0: No NSS pulse
1: NSS pulse generated

*Note:* 1. This bit must be written only when the SPI is disabled (SPE=0).
2. This bit is not used in I2S mode and SPI TI mode.
Bit 2 **SSOE**: SS output enable
- 0: SS output is disabled in master mode and the SPI interface can work in multimaster configuration
- 1: SS output is enabled in master mode and when the SPI interface is enabled. The SPI interface cannot work in a multimaster environment.

*Note: This bit is not used in I²S mode and SPI Ti mode.*

Bit 1 **TXDMAEN**: Tx buffer DMA enable
- When this bit is set, a DMA request is generated whenever the TXE flag is set.
- 0: Tx buffer DMA disabled
- 1: Tx buffer DMA enabled

Bit 0 **RXDMAEN**: Rx buffer DMA enable
- When this bit is set, a DMA request is generated whenever the RXNE flag is set.
- 0: Rx buffer DMA disabled
- 1: Rx buffer DMA enabled

### 34.9.3 SPI status register (SPIx_SR)

**Address offset:** 0x08  
**Reset value:** 0x0002

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>rc_w0</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

- Bits 15:13 Reserved, must be kept at reset value.
- Bits 12:11 **FTLV[1:0]**: FIFO transmission level
  - These bits are set and cleared by hardware.
  - 00: FIFO empty
  - 01: 1/4 FIFO
  - 10: 1/2 FIFO
  - 11: FIFO full (considered as FULL when the FIFO threshold is greater than 1/2)

*Note: This bit is not used in I²S mode.*

- Bits 10:9 **FRLVL[1:0]**: FIFO reception level
  - These bits are set and cleared by hardware.
  - 00: FIFO empty
  - 01: 1/4 FIFO
  - 10: 1/2 FIFO
  - 11: FIFO full

*Note: These bits are not used in PS mode and in SPI receive-only mode while CRC calculation is enabled.*

- Bit 8 **FRE**: Frame format error
  - This flag is used for SPI in TI slave mode and I²S slave mode. Refer to **Section 34.5.11: SPI error flags** and **Section 34.7.8: I²S error flags**.
  - This flag is set by hardware and reset when SPIx_SR is read by software.
  - 0: No frame format error
  - 1: A frame format error occurred
Bit 7  **BSY**: Busy flag
0: SPI (or I2S) not busy
1: SPI (or I2S) is busy in communication or Tx buffer is not empty
This flag is set and cleared by hardware.
*Note:* The BSY flag must be used with caution: refer to Section 34.5.10: SPI status flags and Procedure for disabling the SPI on page 1105.

Bit 6  **OVR**: Overrun flag
0: No overrun occurred
1: Overrun occurred
This flag is set by hardware and reset by a software sequence. Refer to I2S error flags on page 1137 for the software sequence.

Bit 5  **MODF**: Mode fault
0: No mode fault occurred
1: Mode fault occurred
This flag is set by hardware and reset by a software sequence. Refer to Section : Mode fault (MODF) on page 1115 for the software sequence.
*Note:* This bit is not used in I2S mode.

Bit 4  **CRCERR**: CRC error flag
0: CRC value received matches the SPIx_RXCRCR value
1: CRC value received does not match the SPIx_RXCRCR value
*Note:* This flag is set by hardware and cleared by software writing 0.
This bit is not used in I2S mode.

Bit 3  **UDR**: Underrun flag
0: No underrun occurred
1: Underrun occurred
This flag is set by hardware and reset by a software sequence. Refer to I2S error flags on page 1137 for the software sequence.
*Note:* This bit is not used in SPI mode.

Bit 2  **CHSIDE**: Channel side
0: Channel Left has to be transmitted or has been received
1: Channel Right has to be transmitted or has been received
*Note:* This bit is not used in SPI mode. It has no significance in PCM mode.

Bit 1  **TXE**: Transmit buffer empty
0: Tx buffer not empty
1: Tx buffer empty

Bit 0  **RXNE**: Receive buffer not empty
0: Rx buffer empty
1: Rx buffer not empty
34.9.4 SPI data register (SPIx_DR)

Address offset: 0x0C
Reset value: 0x0000

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

Bits 15:0 **DR[15:0]:** Data register
Data received or to be transmitted
The data register serves as an interface between the Rx and Tx FIFOs. When the data register is read, RxFIFO is accessed while the write to data register accesses TxFIFO (See Section 34.5.9: Data transmission and reception procedures).
Note: Data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read. The Rx threshold setting must always correspond with the read access currently used.

34.9.5 SPI CRC polynomial register (SPIx_CRCPR)

Address offset: 0x10
Reset value: 0x0007

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

Bits 15:0 **CRCPOLY[15:0]:** CRC polynomial register
This register contains the polynomial for the CRC calculation.
The CRC polynomial (0x0007) is the reset value of this register. Another polynomial can be configured as required.

Note: The polynomial value should be odd only. No even value is supported.

34.9.6 SPI Rx CRC register (SPIx_RXCRCR)

Address offset: 0x14
Reset value: 0x0000

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

STI
Bits 15:0 **RXCRC[15:0]: Rx CRC register**

When CRC calculation is enabled, the RXCRC[15:0] bits contain the computed CRC value of the subsequently received bytes. This register is reset when the CRCEN bit in SPIx_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the CRC frame format is set to be 8-bit length (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit CRC frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

*Note:* A read to this register when the BSY Flag is set could return an incorrect value.

These bits are not used in I2S mode.

### 34.9.7 SPI Tx CRC register (SPIx_TXCRCR)

Address offset: 0x18

Reset value: 0x0000

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

Bits 15:0 **TXCRC[15:0]: Tx CRC register**

When CRC calculation is enabled, the TXCRC[7:0] bits contain the computed CRC value of the subsequently transmitted bytes. This register is reset when the CRCEN bit of SPIx_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the CRC frame format is set to be 8-bit length (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit CRC frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

*Note:* A read to this register when the BSY flag is set could return an incorrect value.

These bits are not used in I2S mode.

### 34.9.8 SPIx_I2S configuration register (SPIx_I2SCFGR)

Address offset: 0x1C

Reset value: 0x0000

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

1146/1234
Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **ASTRTEN**: Asynchronous start enable.
   0: The Asynchronous start is disabled.
   1: The Asynchronous start is enabled.
   When the I2S is enabled in slave mode, the hardware starts the transfer when the I2S clock is
   received and an appropriate transition is detected on the WS signal.
   When the I2S is enabled in slave mode, the hardware starts the transfer when the I2S clock is
   received and the appropriate level is detected on the WS signal.

*Note: The appropriate transition is a falling edge on WS signal when I2S Philips Standard is used,
   or a rising edge for other standards.*

*The appropriate level is a low level on WS signal when I2S Philips Standard is used, or a high
   level for other standards.*

*Please refer to Section 34.7.3: Start-up description for additional information.*

Bit 11 **I2SMOD**: I2S mode selection
   0: SPI mode is selected
   1: I2S mode is selected

*Note: This bit should be configured when the SPI is disabled.*

Bit 10 **I2SE**: I2S enable
   0: I2S peripheral is disabled
   1: I2S peripheral is enabled

*Note: This bit is not used in SPI mode.*

Bits 9:8 **I2SCFG[1:0]**: I2S configuration mode
   00: Slave - transmit
   01: Slave - receive
   10: Master - transmit
   11: Master - receive

*Note: These bits should be configured when the I2S is disabled.*

*They are not used in SPI mode.*

Bit 7 **PCMSYNC**: PCM frame synchronization
   0: Short frame synchronization
   1: Long frame synchronization

*Note: This bit has a meaning only if I2SSTD = 11 (PCM standard is used).*

*It is not used in SPI mode.*

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I2S standard selection
   00: I²S Philips standard
   01: MSB justified standard (left justified)
   10: LSB justified standard (right justified)
   11: PCM standard

For more details on I²S standards, refer to **Section 34.7.2 on page 1121**

*Note: For correct operation, these bits should be configured when the I2S is disabled.*

*They are not used in SPI mode.*
Bit 3 **CKPOL**: Inactive state clock polarity
- 0: I2S clock inactive state is low level
- 1: I2S clock inactive state is high level

*Note: For correct operation, this bit should be configured when the I2S is disabled.*
- It is not used in SPI mode.
- The bit CKPOL does not affect the CK edge sensitivity used to receive or transmit the SD and WS signals.

Bits 2:1 **DATLEN[1:0]**: Data length to be transferred
- 00: 16-bit data length
- 01: 24-bit data length
- 10: 32-bit data length
- 11: Not allowed

*Note: For correct operation, these bits should be configured when the I2S is disabled.*
- They are not used in SPI mode.

Bit 0 **CHLEN**: Channel length (number of bits per audio channel)
- 0: 16-bit wide
- 1: 32-bit wide

The bit write operation has a meaning only if DATLEN = 00 otherwise the channel length is fixed to 32-bit by hardware whatever the value filled in.

*Note: For correct operation, this bit should be configured when the I2S is disabled.*
- It is not used in SPI mode.

### 34.9.9 SPIx_I2S prescaler register (SPIx_I2SPR)

Address offset: 0x20

Reset value: 0x0002

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

Bits 15:10: Reserved, must be kept at reset value.
Bit 9  **MCKOE**: Master clock output enable
- 0: Master clock output is disabled
- 1: Master clock output is enabled

*Note:* This bit should be configured when the I2S is disabled. It is used only when the I2S is in master mode.
It is not used in SPI mode.

Bit 8  **ODD**: Odd factor for the prescaler
- 0: Real divider value is = I2SDIV * 2
- 1: Real divider value is = (I2SDIV * 2) + 1

Refer to Section 34.7.3 on page 1128.

*Note:* This bit should be configured when the I2S is disabled. It is used only when the I2S is in master mode.
It is not used in SPI mode.

Bits 7:0  **I2SDIV[7:0]**: I2S linear prescaler
- I2SDIV [7:0] = 0 or I2SDIV [7:0] = 1 are forbidden values.
- Refer to Section 34.7.3 on page 1128.

*Note:* These bits should be configured when the I2S is disabled. They are used only when the I2S is in master mode.
They are not used in SPI mode.
34.9.10 SPI/I2S register map

Table 185 shows the SPI/I2S register map and reset values.

| Offset | Register       | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x00   | SPIx_CR1       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x04   | SPIx_CR2       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                | 0   | 0   | 0   | 1   | 1   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x08   | SPIx_SR        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x0C   | SPIx_DR        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x10   | SPIx_CRCPR     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | CRCPOLY[15:0]  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x14   | SPIx_RXCRCR    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | RXCRC[15:0]    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x18   | SPIx_TXCRCR    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | TXCRC[15:0]    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1C   | SPIx_I2SCFGR   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | ASTRTEN        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | I2SMOD         |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | I2SE           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | I2SCFG[1:0]    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | PCMSYNC        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | I2SSTD         |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | CKPOL          |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | DATLEN[1:0]    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | CHLEN          |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x20   | SPIx_I2SPR     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | MCKOE          |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | I2SDIV[7:0]    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | I2SDIV[7:0]    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Refer to Section 2.2 on page 54 for the register boundary addresses.
35.1 Introduction
The USB Type-C / USB Power Delivery interface supports the physical layer of the USB Power Delivery specification.
- Supports Universal Serial Bus Power Delivery specification: Revision 3.0, V1.1, 12 January, 2017

Primary function is the physical layer implementation of the Power Delivery (PD) specification, CC signaling method (not VBUS), so limited to Type-C cables.

35.2 UCPD main features
The UCPD controller is compliant with
- USB Type-C Rev. 1.2 and
- USB Power Delivery Rev. 3.0 specifications

The UCPD controller uses specific I/Os supporting the USB Type-C and USB Power Delivery requirements, featuring:
- USB Type-C pull-up (Rp, all values) and pull-down (Rd) resistors
- “Dead battery” support
- USB Power Delivery message transmission and reception
- FRS (fast role swap) support

The digital controller handles notably
- USB Type-C level detection with de-bounce, generating interrupts
- FRS detection, generating an interrupt
- byte-level interface for USB Power Delivery payload, generating interrupts (DMA compatible)
- USB Power Delivery timing dividers (including a clock pre-scaler)
- CRC generation/checking
- 4b5b encode/decode
- ordered sets (with a programmable ordered set mask at receive)
- frequency recovery in receiver during preamble

The interface offers low-power operation compatible with Stop mode, maintaining the capacity to detect incoming USB Power Delivery messages and FRS signaling.

35.3 UCPD implementation
The STM32G081 family implements two UCPD controllers, allowing support of two USB Type-C ports.
Table 186. UCPD implementation

<table>
<thead>
<tr>
<th>UCPD features</th>
<th>UCPD1</th>
<th>UCPD2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dead battery support via DBCC pins</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FRSTX</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Trimming is fully automatic (no SW override necessary)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Discrete component PHY option support</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. “X” = supported, “-” = not supported

35.4 UCPD functional description

The USB Power Delivery peripheral provides hardware support of the physical layer of USB Power Delivery specification.

Supports:
- USB Power Delivery Revision 3.0, including Fast Role Swap
- Works with ST’s integrated on-chip PHY,
  - Direct detection of Type-C voltage levels
  - Supporting Power Delivery “Fast Role Swap” (Rx side)
- “Fast Role Swap” Tx control generation (if required) for connection to external MOSFETs on CC lines

Key features of UCPD Tx side:
- Byte data reception (from the protocol layer)
- Preamble generation
- 4b5b encoding of payload data
- BMC (bi-phase mark) coding
- Calculation and appending a CRC
- Hardware management of InterFrame gap (Tx message will be delayed to guarantee this)
- Hardware monitoring of the IDLE state on the CC line (Tx message will be discarded if not immediately IDLE)
- Hardware support of “Hard Reset” mechanism (interrupts any on-going Tx message)
- Transmit the packet (preamble, SOP*, payload, CRC and EOP) across the channel using BMC coding over the CC pin of a Type-C USB connector
Key features of the UCPD Rx side:
- Preamble recovery
- BMC (bi-phase mark) decoding
- 5b4b decoding to get payload data
- EOP detection
- CRC checking (based on EOP detection)
- K-Code ordered set detection: Hardware detection of the 5 SOP and 2 Reset sets, plus two software-defined patterns (allows for only 3 out of 4 K-codes being correctly received as defined by the standard)

Other key features:
- Type-C pull-up and pull-down resistor support (all three Rp values and Rd)
- Dead battery Rd support
- Type-C voltage level detection
- Power Delivery BIST modes supported (minimum required for BMC signaling case): BIST carrier mode 2 (Tx only); BIST Test Data (Tx and Rx)

### 35.4.1 UCPD block diagram

The external pins of the UCPD are listed in the Table 187 below:
### Table 187. UCPD pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCPDx_FRSTX</td>
<td>Digital</td>
<td>Universal serial bus Fast Role Swap signaling control, applicable only to DRPs. Control signal (active high) for external NMOS to pull down the active CC line to GND (Fast Role Swap signaling). Typical application will have two such MOS (one for CC1 and one for CC2). Initially both possible mappings should be in GPIO mode driving low. After attach the FRS pulse signal should be remapped by selection of this signal as the active &quot;alternate function&quot; of the GPIO pin (see GPIO chapter). The cable orientation (found through the Type-C state machine will determine the correct CC line to be mapped). The inactive CC pin should be driven with a logic 0 level using GPIO mode.</td>
</tr>
<tr>
<td>UCPDx_CC1</td>
<td>Analog</td>
<td>Universal serial bus Type-C Configuration control line 1. For connection to CC1 line of USB Type-C connector</td>
</tr>
<tr>
<td>UCPDx_CC2</td>
<td>Analog</td>
<td>Universal serial bus Type-C Configuration control line 2. For connection to CC2 line of USB Type-C connector</td>
</tr>
<tr>
<td>UCPDx_DBCC1</td>
<td>Analog</td>
<td>Universal serial bus Type-C Configuration control line 1, dead battery pin. If dead battery support required, UCPDx_DBCC1 must be connected to the CC1 line of USB Type-C connector</td>
</tr>
<tr>
<td>UCPDx_DBCC2</td>
<td>Analog</td>
<td>Universal serial bus Type-C Configuration control line 2, dead battery pin. If dead battery support required, UCPDx_DBCC2 must be connected to the CC2 line of USB Type-C connector</td>
</tr>
</tbody>
</table>

Key internal signals are shown in the following table:

### Table 188. UCPD internal signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nPReset</td>
<td>Digital input</td>
<td>Reset pin</td>
</tr>
<tr>
<td>PClk</td>
<td>Digital input</td>
<td>APB bus clock for registers</td>
</tr>
<tr>
<td>UsbpdClk</td>
<td>Digital input</td>
<td>Kernel clock</td>
</tr>
<tr>
<td>TxDmaReq</td>
<td>Digital output</td>
<td>DMA request (BMC Tx)</td>
</tr>
<tr>
<td>TxDmaAck</td>
<td>Digital input</td>
<td>DMA acknowledge (BMC Tx)</td>
</tr>
<tr>
<td>RxDmaReq</td>
<td>Digital output</td>
<td>DMA request (BMC Rx)</td>
</tr>
<tr>
<td>RxDmaAck</td>
<td>Digital input</td>
<td>DMA acknowledge (BMC Rx)</td>
</tr>
<tr>
<td>IrqRxHrst</td>
<td>Digital output</td>
<td>Interrupt output (specific to Hard Reset) for connection - not used</td>
</tr>
<tr>
<td>IrqOther</td>
<td>Digital output</td>
<td>Interrupt output (other interrupts) - not used</td>
</tr>
<tr>
<td>IrqAll</td>
<td>Digital output</td>
<td>Interrupt output (all interrupts) for connection to NVIC</td>
</tr>
<tr>
<td>IrqAsyn</td>
<td>Digital output</td>
<td>Output for connection to EXTI</td>
</tr>
<tr>
<td>ClkReq</td>
<td>Digital output</td>
<td>Output for connection to RCC</td>
</tr>
</tbody>
</table>
35.4.2 UCPD resets and clocks

A single reset signal nPReset (APB bus reset) is used.

The register section of UCPD is directly clocked on PClk.

The main functional part is clocked on UsbpdClk. It can be pre-scaled to a more appropriate frequency using PSC_USBPDCLK[2:0].

The receiver is designed to work for any clock input from 6 to 18 MHz. Performance may be lower in the range 6 to 9 MHz.

The following diagram shows the main elements relating to the clocks and other timing elements that need to be set to appropriate values.

![Clock division and timing elements](Figure 394. Clock division and timing elements)

Refer to the USB PD specification in order to program the fixed delays. Note that for tTransitionWindow and especially for tInterFrameGap the clock frequency uncertainty should be taken into account in order to respect the timings in all cases.

35.4.3 Physical layer protocol

The physical layer covers the signaling underlying the USB Power Delivery specification. Its main function (transmit side) is to form packets according to the defined packet format including generally:

- Preamble
- Start of packet (ordered set)
- Payload: header
- Payload: data (with encoded data)
- Cyclic redundancy check (CRC) information
- End of packet

A final step of Bi-phase Mark encoding (BMC) is done before transmission on a CC pin.

There are also timing requirements to be met.
On the receive side the principle task is to extract the:
- Start of packet (ordered set) information
- Payload: header
- Payload: data (decoded)

It also checks the correct reception of the:
- CRC
- End of packet

The receiver is basically a reverse process of the transmitter, thus starting with a BMC decoder.

**Symbol encoding**

Apart from the preamble all symbols are encoded with a 4b5b scheme.

Symbols are 4b5b encoded according to the specification shown in Table 189: 4b5b Symbol Encoding Table:

<table>
<thead>
<tr>
<th>Name</th>
<th>4b</th>
<th>5b</th>
<th>Symbol description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>11110</td>
<td>hex data 0</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>01001</td>
<td>hex data 1</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>10100</td>
<td>hex data 2</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>10101</td>
<td>hex data 3</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>01010</td>
<td>hex data 4</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>01011</td>
<td>hex data 5</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>01110</td>
<td>hex data 6</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>01111</td>
<td>hex data 7</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>10010</td>
<td>hex data 8</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>10011</td>
<td>hex data 9</td>
</tr>
<tr>
<td>A</td>
<td>1010</td>
<td>10110</td>
<td>hex data A</td>
</tr>
<tr>
<td>B</td>
<td>1011</td>
<td>10111</td>
<td>hex data B</td>
</tr>
<tr>
<td>C</td>
<td>1100</td>
<td>11010</td>
<td>hex data C</td>
</tr>
<tr>
<td>D</td>
<td>1101</td>
<td>11011</td>
<td>hex data D</td>
</tr>
<tr>
<td>E</td>
<td>1110</td>
<td>11100</td>
<td>hex data E</td>
</tr>
<tr>
<td>F</td>
<td>1111</td>
<td>11101</td>
<td>hex data F</td>
</tr>
<tr>
<td>Sync-1</td>
<td>K-code</td>
<td>11000</td>
<td>Startsynch #1</td>
</tr>
<tr>
<td>Sync-2</td>
<td>K-code</td>
<td>10001</td>
<td>Startsynch #2</td>
</tr>
<tr>
<td>RST-1</td>
<td>K-code</td>
<td>00111</td>
<td>Hard Reset #1</td>
</tr>
<tr>
<td>RST-2</td>
<td>K-code</td>
<td>11001</td>
<td>Hard Reset #2</td>
</tr>
<tr>
<td>EOP</td>
<td>K-code</td>
<td>01101</td>
<td>EOP End Of Packet</td>
</tr>
<tr>
<td>Reserved</td>
<td>Error</td>
<td>00000</td>
<td>Do Not Use</td>
</tr>
</tbody>
</table>
An ordered set consists of 4 K-codes as shown below.

Packet format is shown in Figure 395:

Defined ordered sets are listed in Table 190. This list defines all of the possible SOP* sequences.

At the physical layer, only Hard Reset has a special behavior (Hard Reset has a higher priority so has the ability to interrupt an on-going Tx message).
On reception ordered sets are to be validated as defined in the specification shown in Table 191:

<table>
<thead>
<tr>
<th>Table 190. Ordered sets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ordered set name</strong></td>
</tr>
<tr>
<td>SOP</td>
</tr>
<tr>
<td>SOP'</td>
</tr>
<tr>
<td>SOP''</td>
</tr>
<tr>
<td>Hard Reset</td>
</tr>
<tr>
<td>Cable Reset</td>
</tr>
<tr>
<td>SOP'_Debug</td>
</tr>
<tr>
<td>SOP''_Debug</td>
</tr>
</tbody>
</table>

In summary, the physical layer must accept any combination of 3/4 correct K-codes.

Transmitted bit ordering

Transmitted sizes are defined in the specification shown in Table 192.

<table>
<thead>
<tr>
<th>Table 191. Validation of ordered sets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status</strong></td>
</tr>
<tr>
<td>Valid</td>
</tr>
<tr>
<td>Valid K-code</td>
</tr>
<tr>
<td>Valid K-code</td>
</tr>
<tr>
<td>Valid K-code</td>
</tr>
<tr>
<td>Valid (perfect) K-code</td>
</tr>
<tr>
<td>Not valid (example) K-code</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 192. Data size</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data unit</strong></td>
</tr>
<tr>
<td>Byte</td>
</tr>
<tr>
<td>Word</td>
</tr>
<tr>
<td>DWord</td>
</tr>
</tbody>
</table>

The order used to transmit the bits is shown in Figure 396.
Figure 396. Transmit order for various sizes of data

The packet format is shown below in Figure 397. This also details the encoding, the source of the data (e.g. physical layer / protocol layer).

Figure 397. Packet format

<table>
<thead>
<tr>
<th>Preamble (training for receiver)</th>
<th>SOP* (start of packet)</th>
<th>Header</th>
<th>Byte 0</th>
<th>Byte 1</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Byte n-1</td>
<td>CRC</td>
<td>EOP (end of packet)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LEGEND:
- Training sequence provided by the Physical layer, not encoded with 4b5b
- Provided by the Physical layer, encoded with 4b5b
- Provided by the Protocol layer, encoded with 4b5b
Packet format: Hard Reset (special case)

Among all ordered sets, the "Hard Reset" requires special treatment by the Physical Layer hardware. This is due to the fact that this is a higher priority code and should directly interrupt (in a clean manner) an on-going transfer.

The physical layer specification describes the behavior thus:

3. If the PHY Layer is currently sending a message, the message shall be interrupted by sending an EOP K-code and the rest of the message discarded.
4. Wait tInterFrameGap.
5. If CC is not idle, wait for it to become idle
6. Send the Preamble followed by the 4 K-codes for Hard Reset signaling.
7. Disable the channel (i.e. stop sending and receiving), reset the PHY Layer and inform the Protocol Layer that the PHY Layer has been reset.
8. Re-enable the channel when requested by the Protocol Layer

![Figure 398. Line format of Hard Reset](image)

Packet format: Cable Reset

Cable Reset is similar in format to Hard Reset, but unlike Hard Reset it does not require a specific high priority treatment.

The format is shown below in Figure 399

![Figure 399. Line format of Cable Reset](image)
Collision avoidance

The physical layer respects a delay, “tInterFrameGap”, between end of last transmitted bit of a Tx message, and the first bit of a following message.

It also checks the “idle” state of the CC line before starting transmission. The definition of “idle” is less than “nTransitionCount” (i.e. 3) transitions in “tTransitionWindow” (i.e. 12 to 20 μs). For collision avoidance in PD3.0 it is also necessary to manage Rd value (source) and monitor Type-C voltage level for these Rp modifications (at the sink).

Physical layer signaling schemes

The BMC scheme is implemented.

BIST: General

Depending on the BIST action required by the Protocol Layer, either of the following can be run:

- a Tx BIST pattern test, achieved by writing TXMODE and TXSEND
- an Rx BIST pattern test, achieved by writing RXMODE to the correct value for RXBIST.

The two possible patterns supported in UCPD (corresponding to “BMC” mode) are:

- BIST Test Data (192 bit pattern), applies to Tx and Rx. In the case of Rx, the message will be received (but discarded rather than passing to the protocol layer, which must nevertheless still generate a GoodCRC Tx message in acknowledgment).
- BIST Carrier Mode 2 (single pattern, infinite length message), applies to Tx only, the Rx opposite a Tx in this mode should simply ignore the CC line during this state.

BIST pattern option 1: Test Data pattern

Note that the Test Data pattern is not viewed as a special case in UCPD.

The BIST Test Data packet frame format is shown in Figure 400:

![Figure 400. BIST Test Data Frame](image)

This is a fixed length test data pattern. In reality the only aspect that marks its difference from the general packet format already shown in Figure 397: Packet format is the contents of the Header. As UCPD receives the Tx Header contents via programming (it is simply
viewed as part of the payload), it is only this programming (and not the block's behaviour) that differentiates the general packet from the BIST Test Data packet.

**BIST option 2: Carrier Mode 2**

When required, this BIST test mode will send an alternating pattern of 1010 that is continually repeated. As this mode is intended for signal analysis it is stable condition with (in V1.0 of the USB PD specification) no defined length. Starting from V1.1 of the USB PD specification the protocol layer defines a counter that defines when this mode should be turned off.

Note that to quit this sending (according to requirements of the USB PD specification) UCPD should be disabled via UCPDEN after the appropriate delay.

![Figure 401. BIST Carrier Mode 2 Frame](image)

**35.4.4 UCPD BMC Transmitter**

The BMC (biphase mark coding) transmitter is responsible for generating correct Tx messages, respecting timing, and includes functions for the following aspects of the Tx message content:

- 4b5b encoding
- CRC generation
- Biphase Mark Coding (BMC)

The transmitter architecture is shown below in *Figure 402*:
BMC encoder

The method is defined by an IEC specification:

  [www.iec.ch](http://www.iec.ch)

UCPD generates bits based on a simple (integral) clock divider giving all bits the same length (excluding second order jitter issues). The half-bit time will be nominally exactly half the bit time, and only second order effect of rising/falling edge mismatch in the SoC will make a small deviation between high and low times for half-bits.

The field controlling the timing for a half-bit is **HBITCLKDIV**.

Transmitter Timing and collision avoidance

Hardware support of collision avoidance is made as a function of the half bit time for the transmitter. Two counters are implemented:

- tInterFrameGap: via IFRGAP (pre-defined value, can be altered)
- tTransitionWindow: via TRANSWIN (pre-defined value, can be altered)

These two counters once set correctly will generate the interframe gap.

Hard Reset in Transmitter

In order to facilitate generation of a Hard Reset, a special code of TXMODE field is used. No other fields need to be written.

On writing the correct code, the hardware will force Hard Reset Tx under the correct (optimal) timings with respect to an on-going Tx message, which (if still in progress) will be cleanly terminated by truncating the current sequence and directly appending an EOP K-Code sequence. No specific interrupt will be generated relating to this truncation event.
Transmitter behaviour in the case of errors

The under-run condition (TXUND interrupt) may happen by accident and in this case, the UCPD will be starved of (the correct) Tx payload and will not be able to complete the Tx message correctly. This is a serious error (for this to happen the software will have failed to respond in time). As a result the hardware will ensure the CRC is incorrect at the end of the message, thus guaranteeing the message to be discarded at the receiver.

35.4.5 UCPD BMC Receiver

The UCPD BMC receiver is responsible for these primary functions:
- Preamble detection / timing derivation
- BMC decoding
- 4b5b decoding
- K-Code ordered set recognition (3 of 4 correct basis)
- CRC checking
- SOP detection
- EOP detection (should be always active even ahead of normal behavior due to Hard Reset characteristics)

The receiver is activated as soon as the UCPD is enabled (via UCPDEN), but is inactive during all periods of transmit. At the moment of the enable, in order to start cleanly, it will initially wait for an IDLE line state before attempting to receive a message.

UCPD Receiver high level architecture is shown in Figure 403.
**UCPD Receiver: CRC checker**

The received bits are fed into a CRC checker which evolves a 32-bit state during the received the payload bitstream. At the end the 32 bits of the CRC also fed into the logic.

The EOP detection (5 bits) halts the process and a check is performed for the fixed residual state which confirms correct reception of the payload (in fact the residual is 0xC704DD78).

At this point the UCPD raises interrupt RXMSGEND. If the CRC was not correct then RXERR will be set true and the receive data should be discarded.

Under normal operation, this interrupt would previously have been acknowledged and thus cleared. If this is not the case, a different interrupt RXOVR will be generated in place of RXMSGEND.

**UCPD Receiver: Ordered set detection**

This function will detect the different ordered sets each consisting of four five-bit K-Codes.

Once we are in the preamble we will open a sliding window detection of the ordered set (4 words of 5 bits).
The ordered sets detected will thus include all SOP* codes:
- SOP
- SOP'
- SOP''
but also
- Hard Reset
- Cable Reset
- SOP'_Debug
- SOP''_Debug
- Two extensions defined by registers USBPD1_RX_ORDEXT1 and USBPD1_RX_ORDEXT2

**UCPD Receiver: EOP detection and Hard Reset exception handling**

EOP is a fixed 5-bit K-Code marking the end of a message.

The way in which a transmitter is required to send a Hard Reset (should a previous message transmit still be in progress) is that this previous message is truncated early with an EOP.

If Hard Reset were ignored, then the EOP detection could be done only at the expected time. However, due to the Hard Reset issue, the EOP detector must be active while an Rx message is arriving. When an “early EOP” is detected, the truncated Rx message will immediately be discarded.

**UCPD Receiver exception handling: truncated or corrupt message exception**

Once the ordered set has been detected, depending on the message, there may be data bytes to be received which will be completed with a CRC and EOP. If at any point during these phases, an error condition happens e.g.
- the line becomes static for a time significantly over the “1 UI” length (the exact threshold for this condition is not critical but the exception should happen before 3 UIs),
- the message goes to the end but it is not recognized (e.g. EOP is corrupted)

in both of these cases, the receiver quits the current message, raising RXMSGEND with RXERR at 1.

**UCPD Receiver exception handling: short preamble or incomplete ordered set exception**

In the exceptional case of the receiver seeing less that half of the expected preamble, the frequency estimation allowing correct BMC-decode becomes impossible. Even if the full preamble is seen, allowing frequency estimation, but the ordered set is not fully received before the line becomes static, the receiver state machine will not start.

In both of these cases, the clock-recovery/BMC decoder will re-start, checking initially for an IDLE condition, followed by a preamble, and then estimating frequency.
35.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd)

UCPD offers simple control of these resistors via ANAMODE[1:0] and ANASUBMODE[1:0]. In case only one of the CC pins is to be used, it is possible to optimize power consumption by disabling control on one of the pins using CCENABLE[1:0].

When the MCU is unpowered, it will still present the "dead battery" Rd, provided that UCPDx_DBCC1 and UCPDx_DBCC2 pins are each connected to UCPDx_CC1 and UCPDx_CC2 pins respectively. After power arrives and the MCU boots, the desired behavior (for example sink) should be programmed into ANAMODE and ANASUBMODE before setting the UCPDx_STROBE bits of the SYSCFG1_CFGR1 register to activate this behavior.

If dead battery behavior is not required (e.g. for source only products), then UCPDx_DBCC1 and UCPDx_DBCC2 pins should both be tied to ground. After power arrives and the MCU boots, the desired behavior in this case (e.g. source) should be programmed into ANAMODE and ANASUBMODE before setting the UCPDx_STROBE bits of the SYSCFG1_CFGR1 register to activate this behavior.

Use of Standby low-power mode is possible for sinks in the unattached state.

35.4.7 UCPD Type-C voltage monitoring and de-bouncing

In order to keep the Type-C state machine up to date, monitoring of significant voltage events on the CC1 and CC2 pins (corresponding to those state machines) is available and can be done, either continuously or by polling.

In order to present only significant events to software, a de-bouncing is performed, also a co-ordination with power delivery Tx/Rx activity.

The static level on the CC pins is determined based on those settings via threshold detectors on the PHY to give a voltage range value in registers TYPEC_VSTATE_CC1/2[1:0], facilitating the Type-C state machine implementation in software, and also allowing cable orientation to be determined (PHYCCSEL still needs to be set accordingly in order to use the correct pin between CC1 and CC2 for Power Delivery signaling). De-bouncing is done to prevent rapid transitions not representing a Type-C (DC level) change e.g. during power delivery messaging.

It is recommended to use polling (Type-C detectors OFF between polls, only turned on for the poll) rather than wakeup from STOP (which requires Type-C detectors permanently ON). This will optimize power consumption.

35.4.8 UCPD Fast Role Swap (FRS) signaling and detection

FRS signaling

If FRS signaling is required, external N-MOS transistors are required in order to get the low resistance pull down to GND on the appropriate CC line.

To signal the Fast Role Swap (FRS) condition, a 1 should be written to FRSTX. This will generate the condition and the hardware will guarantee the length. A control signal (FRSTX) will pulse (at 1) for the correct duration.
FRS detection
FRS monitoring is enabled by bit FRSRXEN, and this should only be enabled after the correct value of PHYCCSEL is written (this is a result of establishing the CC position dependent on cable orientation).

35.4.9 UCPD DMA Interface
DMA is implemented in the UCPD and when it is enabled the byte-level interrupts to handle USBPD1_TXDR and USBPD1_RXDR registers (Tx and Rx data register, each one byte) are no longer needed.

By enabling bits TXDMAEN and/or RXDMAEN, DMA can be activated independently for Tx and/or Rx functionality.

35.4.10 Wakeup from STOP
For power consumption optimization it is useful to be able to use STOP mode and wait for events on the CC pins to wake up the MCU to RUN.

In order for this to work, it must be first enabled by writing a 1 to WUPEN.

The events causing the wakeup can be:
- Events on the BMC receiver (RXORDDET ; RXHRSTDET), hardware enable PHYRXEN
- Event on the FRS detector (FRSEVT), hardware enable FRSRXEN
- Events on the Type-C detectors (TYPECEVT1 ; TYPECEVT2), hardware enables CC1TCDIS; CC2TCDIS

35.4.11 UCPD programming sequences
The following example illustrates the normal use of this UCPD.

- Configure UCPD
- Enable UCPD

In parallel start two processes:
- [On demand from Protocol Layer] Send Tx message
- Poll (or wait for interrupt) concerning Rx messages and recover detail to hand off to Protocol Layer

Repeat the above forever.

Initialization phase
Use the following sequence for a clean startup:
- Prepare all clock divider values initially by writing register UCPD_CFG
- Enable the block via UCPDEN

Type-C state machine handling
For the general application cases of source, sink (and also dual role port which can switch between those two) the software must implement the relevant USB Type-C state machine.

The basic coding is shown below in Table 193: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx:
A further field CCENABLE is provided in order to disable any pull-up/pull-downs on one of the CC pins.

Note: The Type-C state machine depends not only on CC pin levels, but also on VBUS presence detection (sink mode) and when in source mode determines VCONN generation and VBUS state (ON/OFF/+voltage level); discharge. UCPD does not directly control VBUS generation circuitry nor VCONN load switch (enabling VCONN supply generator to be connected to the CC pin), but the application will need these inputs and controls to function correctly.

General programming sequence (the UCPD will previously have been configured i.e. CFG registers, and then enabled)

- First program ANAMODE ; ANASUBMODE based on the current position in USB Type-C state machine (and also the current advertisement in the case of a source). This will turn on the appropriate pull-ups/pull-downs on the CC pins, and define the voltage levels that the TYPEC_VSTATE fields represent. Note that before programming the PHY is effectively off
- Read TYPEC_VSTATE_CC1/2 to determine the initial Type-C state (e.g. whether the local source is connected to a remote sink)
- In the case of no connection then wait for a connection event
- Assuming a connection is detected and assuming a local Power Delivery functionality is implemented, start sending/receiving Power Delivery messages
- When a new interrupt/event occurs on PHYEVT1/2 indicating a change in stable voltage, re-evaluate the implications and give this input to the Type-C state machine

Case of a source that needs to change between one of the three possible Rp values (Default-USB / 1.5A / 3.0A) and the sink connected to it:

- [Source] Simply reprogram ANASUBMODE[1:0]
- [Sink behaviour from that time] PHYEVT1/2 will occur and the TYPEC_VSTATE1/2 will change accordingly

Programming for a dual-role port (DRP) toggling from source to sink:

- Simply re-program ANAMODE ; ANASUBMODE to start the new behavior

Detailed programming sequence (example):

<table>
<thead>
<tr>
<th>ANAMODE</th>
<th>ANASUBMODE[1:0]</th>
<th>Notes</th>
<th>TYPEC_VSTATE_CCx[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Source</td>
<td>00: Disabled</td>
<td>Disabled</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>01: Default USB Rp</td>
<td>vRa[Def]</td>
<td>vRd[Def]</td>
</tr>
<tr>
<td></td>
<td>10: 1.5A Rp</td>
<td>vRa[1.5]</td>
<td>vRd[1.5]</td>
</tr>
<tr>
<td>1: Sink</td>
<td>xx</td>
<td>vRa</td>
<td>vRd-USB</td>
</tr>
</tbody>
</table>

Table 193. Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx

0: Source
00: Disabled
01: Default USB Rp
10: 1.5A Rp
11: 3.0A Rp
1: Sink
xx

Notes:

- First program ANAMODE ; ANASUBMODE based on the current position in USB Type-C state machine (and also the current advertisement in the case of a source). This will turn on the appropriate pull-ups/pull-downs on the CC pins, and define the voltage levels that the TYPEC_VSTATE fields represent. Note that before programming the PHY is effectively off
- Read TYPEC_VSTATE_CC1/2 to determine the initial Type-C state (e.g. whether the local source is connected to a remote sink)
- In the case of no connection then wait for a connection event
- Assuming a connection is detected and assuming a local Power Delivery functionality is implemented, start sending/receiving Power Delivery messages
- When a new interrupt/event occurs on PHYEVT1/2 indicating a change in stable voltage, re-evaluate the implications and give this input to the Type-C state machine

Case of a source that needs to change between one of the three possible Rp values (Default-USB / 1.5A / 3.0A) and the sink connected to it:

- [Source] Simply reprogram ANASUBMODE[1:0]
- [Sink behaviour from that time] PHYEVT1/2 will occur and the TYPEC_VSTATE1/2 will change accordingly

Programming for a dual-role port (DRP) toggling from source to sink:

- Simply re-program ANAMODE ; ANASUBMODE to start the new behavior
Normal case of USB PD transmit

On reception of a message from the Protocol Layer (i.e. to be sent),
Prepare Tx message contents by writing the following registers:
- UCPD_TX_ORDSET
- UCPD_TX_PAYSZ

The message sending can be triggered by writing bit:
- TXSEND
with the appropriate value for the TXMODE field.

Note that the interrupt TXIS will trigger to indicate availability of the TXDR buffer, and when this is the case, write the next data byte to:

- UCPD_TXDR

Note that the same interrupt will repeat based on the quantity of bytes in the payload.

If all happens correctly, once the CRC has been sent, the interrupt TXMSGSENT will be set.

**Normal case of USB PD receive**

Notification of start of the receive message sequence is triggered by an interrupt on UCPD_SR (bit RXORDDET).

Information is recovered by reading:

- UCPD_RX_SOP (on interrupt RXORDDET)
- UCPD_RXDR (on interrupt RXNE, repeats for each byte)
- UCPD_RXPAYSZ (on interrupt RXMSGEND)

Note that the data previously read from UCPD_RXDR above must be discarded at this point if RXERR is seen to be true.

Assuming CRC to have been OK, the received data will then be handed off to the Protocol Layer.

For debug reasons, it may be desirable to track statistics of how many incorrect K-Codes have been received (this is done only when 3/4 K-Codes were valid as defined in the specification). This is facilitated via fields:

- RXSOP3OF4 (single bit indication of presence of corrupt K-code)
- RXSOPKINVALID (identifies which - if any - in the sequence was corrupt)

**Hard Reset transmission (special case)**

As soon as it is known that a Hard Reset needs to be transmitted, writing to UCPD_CR.TXHRST will force the internal state machine to generate the correct sequence.

Note that the value of UCPD_TX_ORDSET does not need be updated in this precise case (the correct code for Hard Reset is sent by UCPD).

The USB Power Delivery specification definition of Hard Reset requires that in the case of an ongoing message transmission, the Hard Reset takes precedence. In this case UCPD will, for example, truncate the payload of the current message, append EOP to the end.

Note that no notification is available via the registers (e.g. TXMSGSEND). This is justified by the fact that the Hard Reset takes precedence over any previous activity (for which it is therefore no longer important to know if it had completed).

**Use of DMA for transmission**

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the UCPD_CR register.
For each message:
- Prepare the whole message in memory (starting with two header bytes)
- Program the DMA operation with a length corresponding to the two header bytes plus 4*#data objects
- Write TXSEND to initiate the message transfer
- If TXMSGDISC then go back to previous line (TXSEND)
- Wait for DMA transfer complete interrupt (i.e. when last Tx byte written to UCPD)
- Double-check subsequent TXMSGSENT interrupt appears

Use of DMA for reception
DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the UCPD_CR register.
Whenever a Rx message may come
- Program a DMA receive operation (and spare buffer) a little longer than the maximum possible message (length depends on extended message support)
- After receiving RXORDDET, DMA operation should start working in the background
- On reception of RXMSGSEND interrupt, read RXPAYSZ
- Double-check RXPAYSZ vs. the # of DMA Rx bytes (should correspond but DMA read of RXDR will be slightly after RXDR gets last byte)
- Process the DMA Rx buffer
- Prepare next Rx DMA buffer as soon as possible in order to be ready

35.5 UCPD low-power modes

A summary of low-power modes is shown below in Table 195: Effect of low power modes on the UCPD.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect</td>
</tr>
<tr>
<td>Stop</td>
<td>Detection of events (Type-C, BMC Rx, FRS detection) remains operational and can wake up the MCU.</td>
</tr>
<tr>
<td>Standby</td>
<td>UCPD is not operating, and cannot wake up the MCU. Pull-downs will remain active if configured.</td>
</tr>
<tr>
<td>Unpowered</td>
<td>Dead battery pull-downs will remain active.</td>
</tr>
</tbody>
</table>

The UCPD is able to wake up the MCU from Stop mode when it recognizes a relevant event, either:
- Type-C event relating to a change in the voltage range seen on either of the CC pins, visible in TYPEC_VSTATE_CCx
- Power delivery receive message with an ordered set matching those filtered according to RXORDSETEN[8:0], visible by reading RXORDSET

Wakeup from STOP is enabled by setting the WUPEN bit in the UCPD_CFG2 register.
At UCPD level three types of event requiring kernel clock activity may occur during STOP i.e.

- Activity on the analog PHY voltage threshold detectors which could later be confirmed to be a stable change between voltage ranges defined in the Type-C specification
- Activity on Power Delivery BMC receiver (coming from the selected CC pin) which could potentially generate an Rx message event (i.e. RXORDSET) later
- Activity on Power Delivery FRS detector which could potentially generate an FRS signaling detection event (i.e. FRSEVT) later

It order to function correctly with the RCC, the clock request signal will be activated (conditional on WUPEN) when there is asynchronous activity on:

- Type-C voltage threshold detectors (coming from either CC pin)
- The Power Delivery receiver signal (coming from the selected CC pin)
- The FRS detection signal (coming from the selected CC pin)

### 35.6 UCPD interrupts

The table below gives the UCPD interrupts.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Event flag/Interrupt clearing method</th>
<th>Interrupt enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast Role Swap detection event</td>
<td>FRSEVT</td>
<td>Write 1 to FRSEVTCF</td>
<td>FRSEVTIE</td>
</tr>
<tr>
<td>Type C voltage level event (CC2)</td>
<td>TYPECEVT2</td>
<td>Write 1 to TYPECEVT2CF</td>
<td>TYPECEVT2IE</td>
</tr>
<tr>
<td>Type C voltage level event (CC1)</td>
<td>TYPECEVT1</td>
<td>Write 1 to TYPECEVT1CF</td>
<td>TYPECEVT1IE</td>
</tr>
<tr>
<td>Rx message received</td>
<td>RXMSGEND</td>
<td>Write 1 to RXMSGENDCF</td>
<td>RXMSGENDIE</td>
</tr>
<tr>
<td>Rx data overflow interrupt</td>
<td>RXOVR</td>
<td>Write 1 to RXOVRCF</td>
<td>RXOVR</td>
</tr>
<tr>
<td>Rx Hard Reset detect interrupt</td>
<td>RXHRSTDET</td>
<td>Write 1 to RXHRSTDETCF</td>
<td>RXHRSTDETIE</td>
</tr>
<tr>
<td>Rx ordered set (4 K-codes) detected interrupt</td>
<td>RXORDDET</td>
<td>Write 1 to RXORDDETCF</td>
<td>RXORDDETIE</td>
</tr>
<tr>
<td>Receive data register not empty interrupt</td>
<td>RXNE</td>
<td>Read data in UCPD_RXDR</td>
<td>RXNEIE</td>
</tr>
<tr>
<td>Tx data underrun condition interrupt</td>
<td>TXUND</td>
<td>Write 1 to TXUNDCF</td>
<td>TXUNDCIE</td>
</tr>
<tr>
<td>HRST sent interrupt</td>
<td>HRSTSENT</td>
<td>Write 1 to HRSTSENTCF</td>
<td>HRSTSENTIE</td>
</tr>
<tr>
<td>HRST discarded interrupt</td>
<td>HRSTDISC</td>
<td>Write 1 to HRSTDISCCF</td>
<td>HRSTDISCIE</td>
</tr>
<tr>
<td>Transmit message abort interrupt</td>
<td>TXMSGABT</td>
<td>Write 1 to TXMSGABTCF</td>
<td>TXMSGABTIE</td>
</tr>
</tbody>
</table>
When an interrupt from the UCPD is received, then the software has to check what is the source of the interrupt by reading the UCPD_SR register.

Depending on which bit is at 1, the ISR should handle that condition and clear the bit by a write to the appropriate bit in UCPD_ICR.

### 35.7 UCPD registers

#### 35.7.1 UCPD configuration register 1 (UCPD_CFG1)

Address offset: 0x000
Reset value: 0x0000 0000

This register is used for IP configuration.

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>
**Bit 31** **UCPDEN**: USB Power Delivery Block Enable  
This field is modified by software.  
It shall be used to enable the peripheral. When disabled, the block is not functional, no received information on CC will be seen or acted upon. After enabling this bit the block is functional and receive logic will immediately start detection on the CC input. When disabled, the IP will terminate any ongoing activity instantly. This behaviour is important, notably, for proper termination of “Carrier Mode 2” transmission.  
0: Disable UCPD, the kernel behaves as if all control registers are in the reset state (in fact 0)  
1: Enable UCPD. Note that each register including control bits (this done not include the configuration registers) all of them should be re-written to their desired values.

**Bit 30** **RXDMAEN**: DMA reception requests enable  
This field is modified by software.  
This register is static and as such can only be updated when UCPDEN=0.  
0: DMA mode disabled for reception  
1: DMA mode enabled for reception  

**Bit 29** **TXDMAEN**: DMA reception requests enable  
This field is modified by software.  
This register is static and as such can only be updated when UCPDEN=0.  
0: DMA mode disabled for transmission  
1: DMA mode enabled for transmission

**Bits 28:20** **RXORDSETEN[8:0]**: Receiver ordered set detection enable  
This field is modified by software.  
This register is static and as such can only be updated when UCPDEN=0.  
Determines the types of ordered sets that the receiver shall detect.  
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b1xxxxxx</td>
<td>SOP detect enabled</td>
</tr>
<tr>
<td>0bxxxxxx1</td>
<td>SOP' detect enabled</td>
</tr>
<tr>
<td>0bxxxxxx1x</td>
<td>SOP” detect enabled</td>
</tr>
<tr>
<td>0bxxxxxx1xx</td>
<td>Hard Reset detect enabled</td>
</tr>
<tr>
<td>0bxxxx1xxx</td>
<td>Cable Detect reset enabled</td>
</tr>
<tr>
<td>0bxxx1xxxx</td>
<td>SOP’_Debug enabled</td>
</tr>
<tr>
<td>0bxx1xxxxx</td>
<td>SOP”_Debug enabled</td>
</tr>
<tr>
<td>0bx1xxxxxx</td>
<td>SOP extension#1 enabled</td>
</tr>
<tr>
<td>0b1xxxxxxx</td>
<td>SOP extension#2 enabled</td>
</tr>
</tbody>
</table>

**Bits 19:17** **PSC_USBPDCLK[2:0]**: Pre-scaler for UCPD_CLK. Incoming kernel clock will first be pre-scaled according to this setting before any other divisions (e.g. those done by TRANSWIN ; IFRGAP; HBITCLKDIV) This division would be useful for cases where incoming clock is above 18 MHz (and could also be considered in the range of 12-18 MHz, where a division by 2 would give 6-9 MHz)  
This field is modified by software.  
This register is static and as such can only be updated when UCPDEN=0.  
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>Bypass pre-scaling / divide by 1</td>
</tr>
<tr>
<td>0x1</td>
<td>Pre-scale clock by dividing by 2</td>
</tr>
<tr>
<td>0x2</td>
<td>Pre-scale clock by dividing by 4</td>
</tr>
<tr>
<td>0x3</td>
<td>Pre-scale clock by dividing by 8</td>
</tr>
<tr>
<td>0x4</td>
<td>Pre-scale clock by dividing by 16</td>
</tr>
</tbody>
</table>

**Bit 16** **Reserved**
### 35.7.2 UCPD configuration register 2 (UCPD_CFG2)

**Address offset:** 0x004  
**Reset value:** 0x0000 0000

This register is used for the configuration of Rx signal filtering in the UCPD.

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

**Bits 15:11 TRANSWIN[4:0]:**  
This field is modified by software.  
This register is static and as such can only be updated when UCPDEN=0.  
The number of cycles (minus 1) of the half bit clock (see HBITCLKDIV) to achieve a legal transitionWindow (set according to IP clock to define an interval of between 12 and 20 us)  
0x0: Not supported  
0x9: Divide by 10 (recommended value)  
0x1F: Divide by 32 (maximum division)

**Bits 10:6 IFRGAP[4:0]:**  
This field is modified by software.  
This register is static and as such can only be updated when UCPDEN=0.  
Clock divider value is used to generate interframe gap (tInterframeGap) hardware timer  
This bit contains the definition of the clock divider (minus 1) in order to generate tInterframeGap from the IP clock.  
0x0: Not supported  
0x0: Divide by 14 (may be useful if Tx clock is below nominal value in USB PD 2.0 specification)  
0xE: Divide by 15 (ideal value if Tx clock is exactly nominal value in USB PD 2.0 specification)  
0xF: Divide by 16 (may be useful if Tx clock is above nominal value in USB PD 2.0 specification)  
0x1F: Divide by 32 (maximum division)

**Bits 5:0 HBITCLKDIV[5:0]:**  
This field is modified by software.  
This register is static and as such can only be updated when UCPDEN=0.  
Clock divider values is used to generate a half-bit clock.  
This register contains the number of cycles (minus one) at the IP for a half bit clock e.g. program 3 for a bit clock that takes 8 cycles of the IP clock "UCPD_CLK".  
0x0: Divide by 1 to produce HBITCLK  
0x1A: Divide by 27 to produce HBITCLK (recommended value)  
0x3F: Divide by 63 to produce HBITCLK
Bits 31:4 Reserved

Bit 3 **WUPEN**: Wakeup from STOP enable
This field is modified by software.
This register is static and as such can only be updated when UCPDEN=0.
0: Wakeup from STOP disable (disables output UCPD_ASYNC_INT)
1: Wakeup from STOP enable (enables output UCPD_ASYNC_INT)

Bit 2 **FORCECLK**: Controls forcing of the clock request ClkReq.
This field is modified by software.
This register is static and as such can only be updated when UCPDEN=0.
0: Do not force clock request ClkReq
1: Force clock request ClkReq

Bit 1 **RXFILT2N3**: Controls the sampling method for an Rx pre-filter for the BMC decoder. The sampling clock in question is the same clock as the receiver (i.e. after pre-scaler).
This field is modified by software.
This register is static and as such can only be updated when UCPDEN=0.
0: Wait for 3 consistent samples before considering it to be a new level
1: Wait for 2 consistent samples before considering it to be a new level

Bit 0 **RXFILTDIS**: Enables an Rx pre-filter for the BMC decoder. The sampling clock in question is the same clock as the receiver (i.e. after pre-scaler).
This field is modified by software.
This register is static and as such can only be updated when UCPDEN=0.
0: Enable Rx input filtering
1: Disable Rx input filtering

### 35.7.3 UCPD control register (UCPD_CR)

Address offset: 0x00C
Reset value: 0x0000 0000
This register is used for controlling the UCPD.

![UCPD_control_register_diagram](image)
Bits 31:22  Reserved, must be kept at reset value.

Bit 21 **CC2TCDIS**: The bit allows the Type-C detector for CC2 to be disabled.
0: Type-C detectors for CC2 enabled according to ANAMODE and ANASUBMODE settings
1: Type-C detectors for CC2 disabled

Bit 20 **CC1TCDIS**: The bit allows the Type-C detector for CC1 to be disabled.
0: Type-C detectors for CC1 enabled according to ANAMODE and ANASUBMODE settings
1: Type-C detectors for CC1 disabled

Bit 19  Reserved, must be kept at reset value.

Bit 18 **RDCH**: Drive "Rdch" condition as required during state "UnattachedWait.SRC" (described in ECN "USB Type-C ECN for Source VCONN Discharge")
This register can only be updated when UCPDEN=1.
0: Normal condition
1: Must be maintained at one during the Source-only state "UnattachedWait.SRC" to respect the Type-C state. Rdch will be driven on the CC pin associated with VCONN (based on PHYCCSEL). Correct usage of this value implies that the CCENABLE setting is not in contradiction. Note that after writing this bit to a new value, the UCPDx_STROBE bit of SYSCFG_CFRGR1 must be set.

Bit 17 **FRSTX**: Signal Fast Role Swap request.
This register can only be updated when UCPDEN=1.
0: Normal stable state
1: Write one to enable start of Fast Role Swap signaling. The bit will be cleared subsequently by hardware, after a delay respecting the USB Power Delivery Revision 3.0.

Bit 16 **FRSRXEN**: Enable FRS request detection function which monitors the state of the CC line (according to PHYCCSEL, which should be correctly set first), in order to signal an FRS Rx event (FRSEVT). This register can only be updated when UCPDEN=1.
0: Do not enable the detection (for example when attached to a non-FRS capable source/sink)
1: Enable FRS detection

Bit 15  Reserved, must be kept at reset value.

Bit 14  Reserved, must be kept at reset value.

Bit 13  Reserved, must be kept at reset value.

Bit 12  Reserved, must be kept at reset value.

Bits 11:10  **CCENABLE[1:0]**: Controls whether for pull-ups and pull-downs controls related to ANAMODE and ANASUBMODE should be applied to CC1 and CC2 analog PHYs.
This register can only be updated when UCPDEN=1.
0x0: Neither PHY is activated (e.g. disabled state of source)
0x1: Controls apply to only CC1 (e.g. when CC2 is driven by VCONN via external VCONN switch)
0x2: Controls apply to only CC2 (e.g. when CC1 is driven by VCONN via external VCONN switch)
0x3: Controls apply to both CC1 and CC2 (normal usage for sink/source)

Bit 9  **ANAMODE**: Analog PHY working mode (use for CC1 and CC2 depends on CCENABLE).
Refer to Table 13: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx
This register can only be updated when UCPDEN=1. After changing this value, UCPDx_STROBE must be set for it to be validated.
0: Source
1: Sink
Bits 8:7 **ANASUBMODE[1:0]**: Analog PHY sub-mode. Refer to Table13: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx. This register can only be updated when UCPDEN=1. Coding depends on the context of ANAMODE.

Bit 6 **PHYCCSEL**: Selects between use of CC1 and CC2 pins for USB Power Delivery signaling, according to the cable orientation discovered at attach. This register can only be updated when UCPDEN=1.
- 0: Use CC1 IO for power delivery communication
- 1: Use CC2 IO for power delivery communication

Bit 5 **PHYRXEN**: Controls enable of USB Power Delivery receiver. Only one of CC1 and CC2 receivers will be enabled based on PHYCCSEL. This register can only be updated when UCPDEN=1.
- 0: Disable receivers
- 1: Enable only receiver selected by PHYCCSEL

Bit 4 **RXMODE**: Receiver mode
Determine the mode of the Receiver.
This register can only be updated when UCPDEN=1.
- 0: Normal receive mode
- 1: BIST receive mode (BIST Test Data Mode). RXORDSET behaves normally, RXDR no longer receives bytes yet the CRC checking still proceeds as a normal message.

Bit 3 **TXHRST**: Command to send a Tx Hard Reset.
This register can only be updated when UCPDEN=1.
- 0: Writing zero will not cause any action in the hardware.
- 1: Write one to trigger the process of Tx Hard Reset message. The bit will be cleared subsequently by hardware, once the message starts to get sent (or is discarded).

Bit 2 **TXSEND**: Command to send a Tx packet. Only one should be written to this bit.
This register can only be updated when UCPDEN=1.
- 0: Writing zero will not cause any action in the hardware.
- 1: Write one to trigger the process of Tx message sending. The bit will be cleared subsequently by hardware, once the message starts to get sent (or is discarded).

Bits 1:0 **TXMODE[1:0]**: Type of Tx packet.
This field is modified by software.
Other values: invalid
- 0x0: Writing this value will initiate the transfer of a Tx message previously defined in other registers.
- 0x1: Writing this value will trigger a the transfer of a Cable Reset sequence
- 0x2: Writing this value will trigger a BIST test sequence send (BIST Carrier Mode 2). From V1.1 of the USB PD specification, there is a counter defined for the duration of this mode. To quit this mode correctly (after the delay "tBISTContMode"), the method proposed is to write UCPDEN=0

### 35.7.4 UCPD Interrupt Mask Register (UCPD_IMR)
Address offset: 0x010
Reset value: 0x0000 0000
This register is used for masking the interrupts.
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved

Bit 20 **FRSEVTIE**: Enable FRSEVT interrupt
This register can only be updated when UCPDEN=1.

Bits 19:16 Reserved

Bit 15 **TYPECEVT2IE**: Enable TYPECEVT2 interrupt
This register can only be updated when UCPDEN=1.

Bit 14 **TYPECEVT1IE**: Enable TYPECEVT1 interrupt
This register can only be updated when UCPDEN=1.

Bit 13 Reserved

Bit 12 **RXMSGENDIE**: Enable RXMSGEND interrupt
This register can only be updated when UCPDEN=1.

0: Disabled
1: Enabled

Bit 11 **RXOVRIE**: Enable RXOVR interrupt
This register can only be updated when UCPDEN=1.

0: Disabled
1: Enabled

Bit 10 **RXHRSTDETIE**: Enable RXHRSTDET interrupt
This register can only be updated when UCPDEN=1.

0: Disabled
1: Enabled

Bit 9 **RXORDDETIE**: Enable RXORDDET interrupt
This register can only be updated when UCPDEN=1.

0: Disabled
1: Enabled

Bit 8 **RXNEIE**: Enable RXNE interrupt
This register can only be updated when UCPDEN=1.

0: Disabled
1: Enabled

Bit 7 Reserved
35.7.5 **UCPD Status Register (UCPD_SR)**

Address offset: 0x014

Reset value: 0x0000 0000

This register is used for status.

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXISIE</td>
<td>TXMSGDISCIE</td>
<td>TXMSGSENTIE</td>
<td>TXMSGABTIE</td>
<td>HRSTDISCIE</td>
<td>HRSTSENTIE</td>
<td>TXUNDIE</td>
<td></td>
</tr>
<tr>
<td>0: Disabled</td>
<td>0: Disabled</td>
<td>0: Disabled</td>
<td>0: Disabled</td>
<td>0: Disabled</td>
<td>0: Disabled</td>
<td>0: Disabled</td>
<td></td>
</tr>
<tr>
<td>1: Enabled</td>
<td>1: Enabled</td>
<td>1: Enabled</td>
<td>1: Enabled</td>
<td>1: Enabled</td>
<td>1: Enabled</td>
<td>1: Enabled</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FRSEVT</th>
<th>TYPEC_VSTATE_CC2[1:0]</th>
<th>TYPEC_VSTATE_CC1[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 6 **TXUNDIE**: Enable TXUND interrupt
This register can only be updated when UCPDEN=1.
0: Disabled
1: Enabled

Bit 5 **HRSTSENTIE**: Enable HRSTSENT interrupt
This register can only be updated when UCPDEN=1.
0: Disabled
1: Enabled

Bit 4 **HRSTDISCIE**: Enable HRSTDISC interrupt
This register can only be updated when UCPDEN=1.
0: Disabled
1: Enabled

Bit 3 **TXMSGABTIE**: Enable TXMSGABT interrupt
This register can only be updated when UCPDEN=1.
0: Disabled
1: Enabled

Bit 2 **TXMSGSENTIE**: Enable TXMSGSENT interrupt
This register can only be updated when UCPDEN=1.
0: Disabled
1: Enabled

Bit 1 **TXMSGDISCIE**: Enable TXMSGDISC interrupt
This register can only be updated when UCPDEN=1.
0: Disabled
1: Enabled

Bit 0 **TXISIE**: Enable TXIS interrupt
This register can only be updated when UCPDEN=1.
0: Disabled
1: Enabled
Bits 31:21 Reserved

Bit 20 **FRSEVT**: Fast Role Swap detection event. It is cleared by writing 1 to FRSEVTCF.

0: No new event
1: A new FRS (Fast Role Swap) receive event has happened on the port

Bits 19:18 **TYPEC_VSTATE_CC2[1:0]**:
This status shows the DC level seen on the CC2 pin, meaning that the voltage level indicated corresponds to a stable state of the pin. In other words, during USB PD messages, the BMC PHY modulation will not leave a DC level on the pin, but the value of this register will be unchanged.

0x0: Lowest voltage state
0x1: Second voltage state
0x2: Third voltage state
0x3: Highest voltage state

Bit 15 **TYPECEVT2**: Type C voltage level event on CC2 pin. It happens whenever **TYPEC_VSTATE_CC2** value changes indicating a new stable voltage on that pin, It is cleared by writing 1 to TYPECEVT2CF.

0: No new event
1: A new event on Type-C monitoring for CC2 pin

Bit 14 **TYPECEVT1**: Type C voltage level event on CC1 pin. It happens whenever **TYPEC_VSTATE_CC1** value changes indicating a new stable voltage on that pin, It is cleared by writing 1 to TYPECEVT1CF.

0: No new event
1: A new event on Type-C monitoring for CC1 pin

Bit 13 **RXERR**: Receive message not completed OK (incorrect CRC, or truncated message e.g. line becomes static before EOP recognized) It is not an interrupt status bit. It is set to the appropriate value whenever RXMSGEND is set true.

0: Last Rx message declared (via RXMSGEND) has been correctly received
1: Last Rx message declared (via RXMSGEND) has been received with errors

Bit 12 **RXMSGEND**: Rx message received (this status becomes true regardless of CRC value being true). This is not true for a Hard Reset message receive (see RXHRSTDET below). This bit indicates that a message has been received. If the RXERR is set, the message is incorrect/must be discarded. It is cleared by writing 1 to RXMSGENDCF.

0: No new Rx message has been received (since last clear)
1: A new Rx message has been received (since last clear)
Bit 11 **RXOVR**: Rx data overflow interrupt
   This bit detects an error condition whereby the buffer of Rx bytes has overrun (not read in time to free space for the incoming byte). It is cleared by writing 1 to RXOVRCF.
   0: No Rx data overrun has occurred
   1: An Rx data overrun has occurred

Bit 10 **RXHRSTDET**: Rx Hard Reset detect interrupt
   This bit signals the reception of a Hard Reset message. It is cleared by writing 1 to RXHRSTDETCF.
   0: No Rx "Hard Reset" message has been correctly received
   1: An Rx "Hard Reset" message has been correctly received

Bit 9 **RXORDDET**: Rx ordered set (4 K-codes) detected interrupt
   This bit signals that an ordered set has been detected. The relevant information is stored in field RXORDSET of register UCPD_RX_ORDSET. It is cleared by writing 1 to RXORDDETCF.
   0: No ordered set has been detected (since last clear)
   1: A new ordered set has been detected (since last clear)

Bit 8 **RXNE**: Receive data register not empty interrupt
   This bit is set by hardware when the UCPD_RXDR register is not empty. It is cleared when UCPD_RXDR is read.
   0: Rx data register empty
   1: Rx data register not empty

Bit 7 **Reserved**

Bit 6 **TXUND**: Tx data underrun condition interrupt
   This bit detects an error condition whereby the Tx data register (TXDR) has under-run (i.e. data was not written in time for use in Transmit message). It is cleared by writing 1 to TXUNDCF.
   0: No Tx data underrun has occurred (since last clear via TXUNDCF)
   1: A Tx data underrun has occurred (since last clear via TXUNDCF)

Bit 5 **HRSTSENT**: HRST sent interrupt
   This bit signals that the HRST message has been sent (as TXMSGSENT but for Hard Reset). It is cleared by writing 1 to HRSTSENTCF.
   0: No HRST message sent interrupt
   1: HRST message sent interrupt

Bit 4 **HRSTDISC**: HRST discarded interrupt
   This bit signals that the HRST message has been discarded (as TXMSGDISC but for Hard Reset). It is cleared by writing 1 to HRSTDISCCF.
   0: No HRST discarded interrupt
   1: HRST discarded interrupt

Bit 3 **TXMSGABT**: Transmit message abort interrupt
   This bit will be set when a Tx message has been aborted due to a subsequent HRST send request taking priority during transmit. It is cleared by writing 1 to TXMSGABTCF.
   0: No transmit message abort interrupt
   1: Transmit message abort interrupt
35.7.6 **UCPD Interrupt Clear Register (UCPD_ICR)**

Address offset: 0x018

Reset value: 0x0000 0000

This register is used for interrupt flag clearing.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FRSEVTCF</td>
<td>FRS event flag (FRSEVT) clear</td>
</tr>
<tr>
<td>30</td>
<td>TYPECEVT2CF</td>
<td>TypeC event (CC2) flag (TYPECEVT2) clear</td>
</tr>
<tr>
<td>29</td>
<td>TYPECEVT1CF</td>
<td>TypeC event (CC1) flag (TYPECEVT1) clear</td>
</tr>
<tr>
<td>28</td>
<td>RXMSGENDCF</td>
<td>RX msg end flag (RXMSGEND)</td>
</tr>
<tr>
<td>27</td>
<td>RXVRDF</td>
<td>RX VRD flag (RXVRD)</td>
</tr>
<tr>
<td>26</td>
<td>RXHSTDGETCF</td>
<td>RX HST DGET flag (RXHSTDGET)</td>
</tr>
<tr>
<td>25</td>
<td>RXORDDETCF</td>
<td>RX OR DDETECT flag (RXORDDETECT)</td>
</tr>
<tr>
<td>24</td>
<td>TXUNDCF</td>
<td>TX UND flag (TXUND)</td>
</tr>
<tr>
<td>23</td>
<td>HRSTSENTCF</td>
<td>Hard reset sent flag (HRSTSENT)</td>
</tr>
<tr>
<td>22</td>
<td>HRSTDISCCF</td>
<td>Hard reset discard flag (HRSTDISC)</td>
</tr>
<tr>
<td>21</td>
<td>TXMSGABTCF</td>
<td>TX msg abort flag (TXMSGABT)</td>
</tr>
<tr>
<td>20</td>
<td>TXMSGSENTCF</td>
<td>TX msg sent flag (TXMSGSENT)</td>
</tr>
<tr>
<td>19</td>
<td>TXMSGDISCCF</td>
<td>TX msg discard flag (TXMSGDISC)</td>
</tr>
<tr>
<td>18</td>
<td>RXORDDETCF</td>
<td>RX OR DDETECT flag (RXORDDETECT)</td>
</tr>
<tr>
<td>17</td>
<td>RXMSGENDCF</td>
<td>RX msg end flag (RXMSGEND)</td>
</tr>
<tr>
<td>16</td>
<td>TXUNDCF</td>
<td>TX UND flag (TXUND)</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>reserved</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved

**Bit 20 FRSEVTCF**: FRS event flag (FRSEVT) clear
This register can only be updated when UCPDEN=1.
Writing 1 to this bit clears the FRSEVT flag in the UCPD_SR register

**Bits 19:16 Reserved**

**Bit 15 TYPECEVT2CF**: TypeC event (CC2) flag (TYPECEVT2) clear
This register can only be updated when UCPDEN=1.
Writing 1 to this bit clears the TYPECEVT2 flag in the UCPD_SR register

---

**Bit 2 TXMSGSENT**: Transmit message sent interrupt. This is the confirmation at the end of every transmit packet that the packet has been completely sent. In one specific case (subsequent interruption by a Hard Reset which truncates the current message), the bit will not be raised. It is cleared by writing 1 to TXMSGSENTCF.

- 0: Tx message has not (yet) been correctly sent
- 1: Tx message has been correctly sent

**Bit 1 TXMSGDISC**: Transmit message discarded interrupt
Transmit of message was not possible due to receive in progress (or noise on the line). Although the Tx message will not be transmitted if non-IDLE state is true, this bit becomes true only when CC becomes IDLE. This applies to all types of Tx messages, including Hard Reset and BIST cases. It is cleared by writing 1 to TXMSGDISCCF.

- 0: No Tx message has been discarded
- 1: A Tx message has been discarded

**Bit 0 TXIS**: Transmit interrupt status
This bit is set by hardware when the UCPD_TXDR register is empty and it needs to be written (i.e. until the last byte of the payload defined by TXPAYSZ). It is cleared when the next data to be sent is written to the UCPD_TXDR register.

- 0: Tx data register not empty (or empty, but not awaiting data)
- 1: Tx data register empty and needs to be written
Bit 14 **TYPECEVT1CF**: TypeC event (CC1) flag (TYPECEVT1) clear  
This register can only be updated when UCPDEN=1.  
Writing 1 to this bit clears the TYPECEVT1 flag in the UCPD_SR register.

Bit 13 Reserved

Bit 12 **RXMSGENDCF**: Rx message received flag (RXMSGEND) clear  
This register can only be updated when UCPDEN=1.  
Writing 1 to this bit clears the RXMSGEND flag in the UCPD_SR register.

Bit 11 **RXOVRCF**: Rx overflow flag (RXOVR) clear  
This register can only be updated when UCPDEN=1.  
Writing 1 to this bit clears the RXOVR flag in the UCPD_SR register.

Bit 10 **RXHRSTDETCF**: Rx Hard Reset detected flag (RXHRSTDET) clear  
This register can only be updated when UCPDEN=1.  
Writing 1 to this bit clears the RXHRSTDET flag in the UCPD_SR register.

Bit 9 **RXORDDETCF**: Rx ordered set detect flag (RXORDDET) clear  
This register can only be updated when UCPDEN=1.  
Writing 1 to this bit clears the RXORDDET flag in the UCPD_SR register.

Bits 8:7 Reserved

Bit 6 **TXUNDCF**: Tx underflow flag (TXUND) clear  
This register can only be updated when UCPDEN=1.  
Writing 1 to this bit clears the TXUND flag in the UCPD_SR register.

Bit 5 **HRSTSENCF**: Hard reset sent flag (HRSTSENT) clear  
This register can only be updated when UCPDEN=1.  
Writing 1 to this bit clears the HRSTSENT flag in the UCPD_SR register.

Bit 4 **HRSTDISCCF**: Hard reset discarded flag (HRSTDISC) clear  
This register can only be updated when UCPDEN=1.  
Writing 1 to this bit clears the HRSTDISC flag in the UCPD_SR register.

Bit 3 **TXMSGABTCF**: Tx message abort flag (TXMSGABT) clear  
This register can only be updated when UCPDEN=1.  
Writing 1 to this bit clears the TXMSGABT flag in the UCPD_SR register.

Bit 2 **TXMSGSENCF**: Tx message sent flag (TXMSGSENT) clear  
This register can only be updated when UCPDEN=1.  
Writing 1 to this bit clears the TXMSGSENT flag in the UCPD_SR register.

Bit 1 **TXMSGDISCCF**: Tx message discarded flag (TXMSGDISC) clear  
This register can only be updated when UCPDEN=1.  
Writing 1 to this bit clears the TXMSGDISC flag in the UCPD_SR register.

Bit 0 Reserved

### 35.7.7 UCPD Tx Ordered Set Type Register (UCPD_TX_ORDSET)

- **Address offset**: 0x01C
- **Reset value**: 0x0000 0000

This register is used for Tx ordered set.
35.7.8 UCPD Tx Paysize Register (UCPD_TX_PAYSZ)

Address offset: 0x020
Reset value: 0x0000 0000

This register is used for Tx paysize.

Bits 31:20 Reserved

Bits 19-0 TXORDSET[19:0]:
This register can only be updated when UCPDEN=1, except during the transmission of a packet (i.e. between TXSEND/TXHRST written as one and the moment they are cleared). In other cases, the field is not updated. Full 20-bit sequence, consisting of 4 K-codes, each of 5 bits, defining the packet to be transmitted. Contains the 20 data bits for transmission, bit 0 (bit 0 of K-Code1) is the first to be transmitted, bit 19 (bit 4 of K-Code4) is the last.

Bits 31:10 Reserved

Bits 9:0 TXPAYSZ[9:0]: Payload size in bytes.
This register can only be updated when UCPDEN=1.
This field is modified by software, and also by hardware.
Contains the number of bytes remaining to be transmitted as a payload of the Tx message. As will be seen in the examples, the header is considered in this definition to be part of the payload, but CRC is not counted.
Note that as each byte (of payload) is written to UCPD_TXDR, register value will decrement. When it reaches 0, TXIS is not set any more.
0x2: To write for a 2 byte payload (corresponds to a Control Message from the Protocol Layer)
0x6: To write for a 6 byte payload (shortest possible Data Message allowed from the Protocol Layer)
0x1E: To write for a 30 byte payload (longest possible Data Message allowed from the Protocol Layer, not considering extended messages)
0x106: To write for a 262 byte payload (longest possible payload corresponding to extended messages)
0x3FF: Longest possible payload (for future expansion)
35.7.9 **UCPD Tx Data Register (UCPD_TXDR)**

Address offset: 0x024

Reset value: 0x0000 0000

This register is used for Tx data.

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

Bits 31:8 Reserved

Bits 7:0 **TXDATA[7:0]**: 8-bit transmit data

This register can only be updated when UCPDEN=1.

Data byte to transmitted over the USB PD interface.

35.7.10 **UCPD Rx Ordered Set Register (UCPD_RX_ORDSET)**

Address offset: 0x028

Reset value: 0x0000 0000

This register is used for Rx ordered set.

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

r r r
### 35.7.11 UCPD Rx Paysize Register (UCPD_RX_PAYSZ)

Address offset: 0x02C  
Reset value: 0x0000 0000  

This register is used for the Rx paysize.

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

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

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

RXPAYSZ[9:0]  
r
Bits 31:10  Reserved

Bits 9:0  **RXPAYSZ[9:0]**: Rx payload size in bytes. This value is not derived from the header but will increment during reception phase as each byte is made available in UCPD_RXDR. To be read only after RXMSGSEND interrupt (reads during reception may see an erroneous value due to internal resynchronization). As will be seen in the examples, the header is considered in this definition to be part of the payload, but CRC is not counted.

Contains the number of payload bytes received.

This register can only be updated when UCPDEN=1.

0x2: 2 byte payload (corresponds to a Control Message from the Protocol Layer)
0x6: 6 byte payload (shortest possible Data Message allowed from the Protocol Layer)
0x1E: 30 byte payload (longest possible Data Message allowed from the Protocol Layer, excluding extended messages)
0x106: 262 byte payload (longest possible payload corresponding to extended messages)
0x3FF: Maximum supported payload (for future expansion)

### 35.7.12 UCPD Receive Data Register (UCPD_RXDR)

Address offset: 0x030

Reset value: 0x0000 0000

This register is used for Rx data.

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td>RXDATA[7:0]: 8-bit receive data</td>
</tr>
<tr>
<td></td>
<td>Data byte received from the USB PD interface</td>
</tr>
</tbody>
</table>

### 35.7.13 UCPD Rx Ordered Set Extension Register #1 (UCPD_RX_ORDEXT1)

Address offset: 0x034

Reset value: 0x0000 0000

This register is used for Rx ordered set extension #1.
35.7.14 UCPD Rx Ordered Set Extension Register #2
(UCPD_RX_ORDEXT2)

Address offset: 0x038
Reset value: 0x0000 0000

This register is used for Rx ordered set extension #2.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x038</td>
<td>UCPD_RX_ORDEXT2</td>
<td>RXSOPX2[19:16]</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x039</td>
<td>UCPD_RX_ORDEXT2</td>
<td>RXSOPX2[15:0]</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved

Bits 19:0 RXSOPX2[19:0]:
This register is static and as such can only be updated when UCPDEN=0.
Clock divider value is used to generate Interframe gap (tInterframeGap) hardware timerFull 20-bit sequence, consisting of 4 K-codes, each of 5 bits, defining a supplementary ordered set (#1) to be matched in hardware.
Contains the full 20 data bits, bit 0 (bit 0 of K-Code1) is the first to be received, bit 19 (bit 4 of K-Code4) is the last.

35.7.15 UCPD register map

The following table gives the UCPD register map and reset values.

Table 197. UCPD register map and reset values

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | UCPD_CFG1 | UCPDEN | RES | RXDMAEN | TXDMAEN | RXORDSETEN[8:0] | PSC_USBPDCLK[2:0] | TRANSWIN[4:0] | IFRGAP[4:0] | HBITCLKDIV[5:0] | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

ST
| Offset | Register                  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|--------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x004  | UCPD_CFG2                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  |
| 0x008  | Reserved                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  |
| 0x00C  | UCPD_CR                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  |
|        | Reset value              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  |
| 0x010  | UCPD_IMR                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  |
|        | Reset value              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  |
| 0x014  | UCPD_SR                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  |
|        | Reset value              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  |
| 0x018  | UCPD_ICR                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  |
|        | Reset value              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  |
| 0x01C  | UCPD_TX_ORDSET           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | TXORDSET[19:0] |
|        | Reset value              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  |
| 0x020  | UCPD_TX_PAYSZ            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | TXPAYSZ[9:0]   |
|        | Reset value              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  |
| 0x024  | UCPD_TXDR                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | TXDATA[7:0]    |
|        | Reset value              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  |
Table 197. UCPD register map and reset values (continued)

| Offset | Register        | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x28   | UCPD_RX_ORDSET  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2C   | UCPD_RX_PAYSZ   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x30   | UCPD_RXDR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x34   | UCPD_RX_ORDEXT  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 1      | RXSOPX1[19:0]   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | Reset value     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x38   | UCPD_RX_ORDEXT  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 2      | RXSOPX2[19:0]   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | Reset value     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.2 on page 54 for the register boundary addresses.
36 HDMI-CEC controller (CEC)

36.1 Introduction

Consumer electronics control (CEC) is part of HDMI (high-definition multimedia interface) standard as appendix supplement 1. It contains a protocol that provides high-level control functions between various audiovisual products. CEC operates at low speeds, with minimum processing and memory overhead.

The HDMI-CEC controller provides hardware support for this protocol.

36.2 HDMI-CEC controller main features

- Complies with HDMI-CEC v1.4 specification
- 32 kHz CEC kernel with 2 clock source options
  - HSI RC oscillator with fixed prescaler (HSI/488)
  - LSE oscillator
- Works in Stop mode for ultra-low-power applications
- Configurable signal-free time before start of transmission
  - Automatic by hardware, according to CEC state and transmission history
  - Fixed by software (7 timing options)
- Configurable peripheral address (OAR)
- Supports Listen mode
  - Enables reception of CEC messages sent to destination address different from OAR without interfering with the CEC line
- Configurable Rx-tolerance margin
  - Standard tolerance
  - Extended tolerance
- Receive-error detection
  - Bit rising error (BRE), with optional stop of reception (BRESTP)
  - Short bit period error (SBPE)
  - Long bit period error (LBPE)
- Configurable error-bit generation
  - on BRE detection (BREGEN)
  - on LBPE detection (LBPEGEN)
  - always generated on SBPE detection
- Transmission error detection (TXERR)
- Arbitration lost detection (ARBLST)
  - With automatic transmission retry
- Transmission underrun detection (TXUDR)
- Reception overrun detection (RXOVR)
36.3 HDMI-CEC functional description

36.3.1 HDMI-CEC pin

The CEC bus consists of a single bidirectional line that is used to transfer data in and out of the device. It is connected to a +3.3 V supply voltage via a 27 kΩ pull-up resistor. The output stage of the device must have an open-drain or open-collector to allow a wired-AND connection.

The HDMI-CEC controller manages the CEC bidirectional line as an alternate function of a standard GPIO, assuming that it is configured as alternate function open drain. The 27 kΩ pull-up must be added externally to the microcontroller.

To not interfere with the CEC bus when the application power is removed, it is mandatory to isolate the CEC pin from the bus in such conditions. This can be done by using a MOS transistor, as shown on Figure 404.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal type</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| CEC  | Bidirectional | Two states:  
  - 1 = high impedance  
  - 0 = low impedance  
  A 27 kΩ resistor must be added externally. |

36.3.2 HDMI-CEC block diagram

Figure 404. HDMI-CEC block diagram

36.3.3 Message description

All transactions on the CEC line consist of an initiator and one or more followers. The initiator is responsible for sending the message structure and the data. The follower is the recipient of any data and is responsible for setting any acknowledgment bits.

A message is conveyed in a single frame that consists of a start bit followed by a header block and optionally an opcode and a variable number of operand blocks.
All these blocks are made of a 8-bit payload - most significant bit is transmitted first - followed by an end of message (EOM) bit and an acknowledge (ACK) bit.

The EOM bit is set in the last block of a message and kept reset in all others. In case a message contains additional blocks after an EOM is indicated, those additional blocks must be ignored. The EOM bit may be set in the header block to ‘ping’ other devices, to make sure they are active.

The acknowledge bit is always set to high impedance by the initiator so that it can be driven low either by the follower that has read its own address in the header, or by the follower that needs to reject a broadcast message.

The header consists of the source logical address field, and the destination logical address field. Note that the special address 0xF is used for broadcast messages.

### Figure 405. Message structure

![Figure 405. Message structure](MS31004V1)

### Figure 406. Blocks

![Figure 406. Blocks](MS31005V1)

#### 36.3.4 Bit timing

The format of the start bit is unique and identifies the start of a message. It must be validated by its low duration and its total duration.

All remaining data bits in the message, after the start bit, have consistent timing. The high-to-low transition at the end of the data bit is the start of the next data bit except for the final bit where the CEC line remains high.
36.4 Arbitration

All devices transmitting - or retransmitting - a message onto the CEC line must ensure that it has been inactive for a number of bit periods. This signal-free time is defined as the time starting from the final bit of the previous frame and depends on the initiating device and the current status as shown in the table below.

Since only one initiator is allowed at any one time, an arbitration mechanism is provided to avoid conflict when more than one initiator begins transmitting at the same time.

CEC line arbitration starts with the leading edge of the start bit and continues until the end of the initiator address bits within the header block. During this period, the initiator must monitor the CEC line, if whilst driving the line to high impedance it reads it back to 0. Assuming then it has lost arbitration, it stops transmitting and becomes a follower.
A configurable time window is counted before starting the transmission.

In the SFT = 0 configuration, HDMI-CEC performs automatic SFT calculation ensuring compliance with the HDMI-CEC standard:

- 2.5 data bit periods if the CEC is the last bus initiator with unsuccessful transmission
- 4 data bit periods if the CEC is the new bus initiator
- 6 data bit periods if the CEC is the last bus initiator with successful transmission

This is done to guarantee the maximum priority to a failed transmission and the lowest one to the last initiator that completed successfully its transmission.

Otherwise there is the possibility to configure the SFT bits to count a fixed timing value. Possible values are 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5 data bit periods.

### 36.4.1 SFT option bit

In case of SFTOPT = 0 configuration, SFT starts being counted when the start-of-transmission command is set by software (TXSOM=1).

In case of SFTOPT=1, SFT starts automatically being counted by the HDMI-CEC device when a bus-idle or line error condition is detected. If the SFT timer is completed at the time TXSOM command is set then transmission starts immediately without latency. If the SFT timer is still running instead, the system waits until the timer elapses before transmission can start.
In case of SFTOPT = 1 a bus-event condition starting the SFT timer is detected in the following cases:

- In case of a regular end of transmission/reception, when TXEND/RXEND bits are set at the minimum nominal data bit duration of the last bit in the message (ACK bit).
- In case of a transmission error detection, SFT timer starts when the TXERR transmission error is detected (TXERR=1).
- In case of a missing acknowledge from the CEC follower, the SFT timer starts when the TXACKE bit is set, that is at the nominal sampling time of the ACK bit.
- In case of a transmission underrun error, the SFT timer starts when the TXUDR bit is set at the end of the ACK bit.
- In case of a receive error detection implying reception abort, the SFT timer starts at the same time the error is detected. If an error bit is generated, then SFT starts being counted at the end of the error bit.
- In case of a wrong start bit or of any uncodified low impedance bus state from idle, the SFT timer is restarted as soon as the bus comes back to hi-impedance idleness.

### 36.5 Error handling

#### 36.5.1 Bit error

If a data bit - excluding the start bit - is considered invalid, the follower is expected to notify such error by generating a low bit period on the CEC line of 1.4 to 1.6 times the nominal data bit period, i.e. 3.6 ms nominally.

![Figure 411. Error bit timing](MS31010V1)

#### 36.5.2 Message error

A message is considered lost and therefore may be retransmitted under the following conditions:

- a message is not acknowledged in a directly addressed message
- a message is negatively acknowledged in a broadcast message
- a low impedance is detected on the CEC line while it is not expected (line error)

Three kinds of error flag can be detected when the CEC interface is receiving a data bit:

#### 36.5.3 Bit Rising Error (BRE)

BRE (bit rising error): is set when a bit rising edge is detected outside the windows where it is expected (see Figure 412). BRE flag also generates a CEC interrupt if the BREIE=1.
In the case of a BRE detection, the message reception can be stopped according to the BRESTP bit value and an error bit can be generated if BREGEN bit is set.

When BRE is detected in a broadcast message with BRESTP=1 an error bit is generated even if BREGEN=0 to enforce initiator’s retry of the failed transmission. Error bit generation can be disabled by configuring BREGEN=0, BRDNOGEN=1.

### 36.5.4 Short Bit Period Error (SBPE)

SBPE is set when a bit falling edge is detected earlier than expected (see Figure 412). SBPE flag also generates a CEC interrupt if the SBPEIE=1.

An error bit is always generated on the line in case of a SBPE error detection. An Error Bit is not generated upon SBPE detection only when Listen mode is set (LSTN=1) and the following conditions are met:

- A directly addressed message is received containing SBPE
- A broadcast message is received containing SBPE AND BRDNOGEN=1

### 36.5.5 Long Bit Period Error (LBPE)

LBPE is set when a bit falling edge is not detected in a valid window (see Figure 412). LBPE flag also generates a CEC interrupt if the LBPEIE=1.

LBPE always stops the reception, an error bit is generated on the line when LBPEGEN bit is set.

When LBPE is detected in a broadcast message an error bit is generated even if LBPEGEN=0 to enforce initiator’s retry of the failed transmission. Error bit generation can be disabled by configuring LBPEGEN=0, BRDNOGEN=1.

Note: The BREGEN=1, BRESTP=0 configuration must be avoided

![Figure 412: Error handling](image-url)
### Table 199. Error handling timing parameters

<table>
<thead>
<tr>
<th>Time</th>
<th>RXTOL</th>
<th>ms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_s$</td>
<td>x</td>
<td>0</td>
<td>Bit start event.</td>
</tr>
<tr>
<td>$T_1$</td>
<td>1</td>
<td>0.3</td>
<td>The earliest time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>$T_{n1}$</td>
<td>x</td>
<td>0.6</td>
<td>The nominal time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td>$T_2$</td>
<td>0</td>
<td>0.8</td>
<td>The latest time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>$T_{ns}$</td>
<td>x</td>
<td>1.05</td>
<td>Nominal sampling time.</td>
</tr>
<tr>
<td>$T_3$</td>
<td>1</td>
<td>1.2</td>
<td>The earliest time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1.3</td>
<td></td>
</tr>
<tr>
<td>$T_{n0}$</td>
<td>x</td>
<td>1.5</td>
<td>The nominal time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td>$T_4$</td>
<td>0</td>
<td>1.7</td>
<td>The latest time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>$T_5$</td>
<td>1</td>
<td>1.85</td>
<td>The earliest time for the start of a following bit.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2.05</td>
<td></td>
</tr>
<tr>
<td>$T_{nf}$</td>
<td>x</td>
<td>2.4</td>
<td>The nominal data bit period.</td>
</tr>
<tr>
<td>$T_6$</td>
<td>0</td>
<td>2.75</td>
<td>The latest time for the start of a following bit.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2.95</td>
<td></td>
</tr>
</tbody>
</table>
36.5.6 Transmission Error Detection (TXERR)

The CEC initiator sets the TXERR flag if detecting low impedance on the CEC line when it is transmitting high impedance and is not expecting a follower asserted bit. TXERR flag also generates a CEC interrupt if the TXERRIE=1.

TXERR assertion stops the message transmission. Application is in charge to retry the failed transmission up to 5 times.

TXERR checks are performed differently depending on the different states of the CEC line and on the RX tolerance configuration.

**Figure 413. TXERR detection**

**Table 200. TXERR timing parameters**

<table>
<thead>
<tr>
<th>Time</th>
<th>RXTOL</th>
<th>ms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tₕ</td>
<td>x</td>
<td>0</td>
<td>Bit start event.</td>
</tr>
<tr>
<td>T₁</td>
<td>1</td>
<td>0.3</td>
<td>The earliest time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>Tₙ₁</td>
<td>x</td>
<td>0.6</td>
<td>The nominal time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td>T₂</td>
<td>0</td>
<td>0.8</td>
<td>The latest time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>Tₙ₈</td>
<td>x</td>
<td>1.05</td>
<td>Nominal sampling time.</td>
</tr>
<tr>
<td>T₃</td>
<td>1</td>
<td>1.2</td>
<td>The earliest time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1.3</td>
<td></td>
</tr>
</tbody>
</table>
### 36.6 HDMI-CEC interrupts

An interrupt can be produced:
- during reception if a Receive Block Transfer is finished or if a Receive Error occurs.
- during transmission if a Transmit Block Transfer is finished or if a Transmit Error occurs.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rx-Byte Received</td>
<td>RXBR</td>
<td>RXBRIE</td>
</tr>
<tr>
<td>End of reception</td>
<td>RXEND</td>
<td>RXENDIE</td>
</tr>
<tr>
<td>Rx-Overrun</td>
<td>RXOVR</td>
<td>RXOVRIE</td>
</tr>
<tr>
<td>Rx-Bit Rising Error</td>
<td>BRE</td>
<td>BREIE</td>
</tr>
<tr>
<td>Rx-Short Bit Period Error</td>
<td>SBPE</td>
<td>SBPEIE</td>
</tr>
<tr>
<td>Rx-Long Bit Period Error</td>
<td>LBPE</td>
<td>LBPEIE</td>
</tr>
<tr>
<td>Rx-Missing Acknowledge Error</td>
<td>RXACKE</td>
<td>RXACKEIE</td>
</tr>
<tr>
<td>Arbitration lost</td>
<td>ARBLST</td>
<td>ARBLSTIE</td>
</tr>
<tr>
<td>Tx-Byte Request</td>
<td>TXBR</td>
<td>TXBRIE</td>
</tr>
<tr>
<td>End of transmission</td>
<td>TXEND</td>
<td>TXENDIE</td>
</tr>
<tr>
<td>Tx-Buffer Underrun</td>
<td>TXUDR</td>
<td>TXUDRIE</td>
</tr>
<tr>
<td>Tx-Error</td>
<td>TXERR</td>
<td>TXERRIE</td>
</tr>
<tr>
<td>Tx-Missing Acknowledge Error</td>
<td>TXACKE</td>
<td>TXACKEIE</td>
</tr>
</tbody>
</table>

### Table 200. TXERR timing parameters (continued)

<table>
<thead>
<tr>
<th>Time</th>
<th>RXTOL</th>
<th>ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{0}$</td>
<td>x</td>
<td>1.5</td>
</tr>
<tr>
<td>$T_{4}$</td>
<td>0</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.8</td>
</tr>
<tr>
<td>$T_{5}$</td>
<td>1</td>
<td>1.85</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2.05</td>
</tr>
<tr>
<td>$T_{nf}$</td>
<td>x</td>
<td>2.4</td>
</tr>
<tr>
<td>$T_{0}$</td>
<td>0</td>
<td>2.75</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2.95</td>
</tr>
</tbody>
</table>
### 36.7 HDMI-CEC registers

Refer to [Section 1.2 on page 50](#) for a list of abbreviations used in register descriptions.

#### 36.7.1 CEC control register (CEC_CR)

Address offset: 0x00  
Reset value: 0x0000 0000

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

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

| 31:3 Reserved, must be kept at reset value. |
Bit 2  **TXEOM:** Tx End Of Message

The TXEOM bit is set by software to command transmission of the last byte of a CEC message. TXEOM is cleared by hardware at the same time and under the same conditions as for TXSOM.

0: TXDR data byte is transmitted with EOM=0
1: TXDR data byte is transmitted with EOM=1

**Note:** TXEOM must be set when CECEN=1.

TXEOM must be set before writing transmission data to TXDR.

If TXEOM is set when TXSOM=0, transmitted message will consist of 1 byte (HEADER) only (PING message)

Bit 1  **TXSOM:** Tx Start Of Message

TXSOM is set by software to command transmission of the first byte of a CEC message. If the CEC message consists of only one byte, TXEOM must be set before of TXSOM.

Start-Bit is effectively started on the CEC line after SFT is counted. If TXSOM is set while a message reception is ongoing, transmission will start after the end of reception.

TXSOM is cleared by hardware after the last byte of the message is sent with a positive acknowledge (TXEND=1), in case of transmission underrun (TXUDR=1), negative acknowledge (TXACKE=1), and transmission error (TXERR=1). It is also cleared by CECEN=0. It is not cleared and transmission is automatically retried in case of arbitration lost (ARBLST=1).

TXSOM can be also used as a status bit informing application whether any transmission request is pending or under execution. The application can abort a transmission request at any time by clearing the CECEN bit.

0: No CEC transmission is on-going
1: CEC transmission command

**Note:** TXSOM must be set when CECEN=1.

TXSOM must be set when transmission data is available into TXDR.

HEADER first four bits containing own peripheral address are taken from TXDR[7:4], not from CEC_CFGR.OAR which is used only for reception.

Bit 0  **CECEN:** CEC enable

The CECEN bit is set and cleared by software. CECEN=1 starts message reception and enables the TXSOM control. CECEN=0 disables the CEC peripheral, clears all bits of CEC_CR register and aborts any on-going reception or transmission.

0: CEC peripheral is off
1: CEC peripheral is on

### 36.7.2  CEC configuration register (CEC_CFGR)

This register is used to configure the HDMI-CEC controller.

Address offset: 0x04

Reset value: 0x0000 0000

**Caution:** It is mandatory to write CEC_CFGR only when CECEN=0.
Bit 31 **LSTN**: Listen mode
LSTN bit is set and cleared by software.
0: CEC peripheral receives only message addressed to its own address (OAR). Messages addressed to different destination are ignored. Broadcast messages are always received.
1: CEC peripheral receives messages addressed to its own address (OAR) with positive acknowledge. Messages addressed to different destination are received, but without interfering with the CEC bus: no acknowledge sent.

Bits 30:16 **OAR[14:0]**: Own addresses configuration
The OAR bits are set by software to select which destination logical addresses has to be considered in receive mode. Each bit, when set, enables the CEC logical address identified by the given bit position.
At the end of HEADER reception, the received destination address is compared with the enabled addresses. In case of matching address, the incoming message is acknowledged and received. In case of non-matching address, the incoming message is received only in listen mode (LSTN=1), but without acknowledge sent. Broadcast messages are always received.
Example:
OAR = 0b000 0000 0010 0001 means that CEC acknowledges addresses 0x0 and 0x5.
Consequently, each message directed to one of these addresses is received.

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

Bit 8 **SFTOP**: SFT Option Bit
The SFTOPT bit is set and cleared by software.
0: SFT timer starts when TXSOM is set by software
1: SFT timer starts automatically at the end of message transmission/reception.

Bit 7 **BRDNOGEN**: Avoid Error-Bit Generation in Broadcast
The BRDNOGEN bit is set and cleared by software.
0: BRE detection with BRESTP=1 and BREGEN=0 on a broadcast message generates an Error-Bit on the CEC line. LBPE detection with LBPEGEN=0 on a broadcast message generates an Error-Bit on the CEC line
1: Error-Bit is not generated in the same condition as above. An Error-Bit is not generated even in case of an SBPE detection in a broadcast message if listen mode is set.

Bit 6 **LBPEGEN**: Generate Error-Bit on Long Bit Period Error
The LBPEGEN bit is set and cleared by software.
0: LBPE detection does not generate an Error-Bit on the CEC line
1: LBPE detection generates an Error-Bit on the CEC line
**Note**: If BRDNOGEN=0, an Error-bit is generated upon LBPE detection in broadcast even if LBPEGEN=0

Bit 5 **BREGEN**: Generate Error-Bit on Bit Rising Error
The BREGEN bit is set and cleared by software.
0: BRE detection does not generate an Error-Bit on the CEC line
1: BRE detection generates an Error-Bit on the CEC line (if BRESTP is set)
**Note**: If BRDNOGEN=0, an Error-bit is generated upon BRE detection with BRESTP=1 in broadcast even if BREGEN=0
Bit 4 **BRESTP**: Rx-Stop on Bit Rising Error  
The BRESTP bit is set and cleared by software.  
0: BRE detection does not stop reception of the CEC message. Data bit is sampled at 1.05 ms.  
1: BRE detection stops message reception  

Bit 3 **RXTOL**: Rx-Tolerance  
The RXTOL bit is set and cleared by software.  
0: Standard tolerance margin:  
  – Start-Bit: +/- 200 µs rise, +/- 200 µs fall.  
  – Data-Bit: +/- 200 µs rise, +/- 350 µs fall.  
1: Extended Tolerance  
  – Start-Bit: +/- 400 µs rise, +/- 400 µs fall  
  – Data-Bit: +/- 300 µs rise, +/- 500 µs fall  

Bits 2:0 **SFT[2:0]**: Signal Free Time  
SFT bits are set by software. In the SFT=0x0 configuration the number of nominal data bit periods waited before transmission is ruled by hardware according to the transmission history. In all the other configurations the SFT number is determined by software.  

\[ 0x0 \]  
  – 2.5 Data-Bit periods if CEC is the last bus initiator with unsuccessful transmission  
    \( (ARBLST=1, TXERR=1, TXUDR=1 \text{ or } TXACKE=1) \)  
  – 4 Data-Bit periods if CEC is the new bus initiator  
  – 6 Data-Bit periods if CEC is the last bus initiator with successful transmission \( (TXEOM=1) \)  

\[ 0x1 \]  
  0.5 nominal data bit periods  
\[ 0x2 \]  
  1.5 nominal data bit periods  
\[ 0x3 \]  
  2.5 nominal data bit periods  
\[ 0x4 \]  
  3.5 nominal data bit periods  
\[ 0x5 \]  
  4.5 nominal data bit periods  
\[ 0x6 \]  
  5.5 nominal data bit periods  
\[ 0x7 \]  
  6.5 nominal data bit periods
36.7.3 **CEC Tx data register (CEC_TXDR)**

Address offset: 0x8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td>TXD[7:0]: Tx data</td>
</tr>
<tr>
<td></td>
<td>TXD is a write-only register containing the data byte to be transmitted.</td>
</tr>
</tbody>
</table>

36.7.4 **CEC Rx data register (CEC_RXDR)**

Address offset: 0xC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td>RXD[7:0]: Rx data</td>
</tr>
<tr>
<td></td>
<td>RXD is read-only and contains the last data byte which has been received from the CEC line.</td>
</tr>
</tbody>
</table>

36.7.5 **CEC Interrupt and Status Register (CEC_ISR)**

Address offset: 0x10
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td>RXD[7:0]: Rx data</td>
</tr>
<tr>
<td></td>
<td>RXD is read-only and contains the last data byte which has been received from the CEC line.</td>
</tr>
</tbody>
</table>
Bits 31:13  Reserved, must be kept at reset value.

Bit 12  **TXACKE**: Tx-Missing Acknowledge Error
In transmission mode, TXACKE is set by hardware to inform application that no acknowledge was received. In case of broadcast transmission, TXACKE informs application that a negative acknowledge was received. TXACKE aborts message transmission and clears TXSOM and TXEOM controls.
TXACKE is cleared by software write at 1.

Bit 11  **TXERR**: Tx-Error
In transmission mode, TXERR is set by hardware if the CEC initiator detects low impedance on the CEC line while it is released. TXERR aborts message transmission and clears TXSOM and TXEOM controls.
TXERR is cleared by software write at 1.

Bit 10  **TXUDR**: Tx-Buffer Underrun
In transmission mode, TXUDR is set by hardware if application was not in time to load TXDR before of next byte transmission. TXUDR aborts message transmission and clears TXSOM and TXEOM control bits.
TXUDR is cleared by software write at 1.

Bit 9  **TXEND**: End of Transmission
TXEND is set by hardware to inform application that the last byte of the CEC message has been successfully transmitted. TXEND clears the TXSOM and TXEOM control bits.
TXEND is cleared by software write at 1.

Bit 8  **TXBR**: Tx-Byte Request
TXBR is set by hardware to inform application that the next transmission data has to be written to TXDR. TXBR is set when the 4th bit of currently transmitted byte is sent. Application must write the next byte to TXDR within 6 nominal data-bit periods before transmission underrun error occurs (TXUDR).
TXBR is cleared by software write at 1.

Bit 7  **ARBLST**: Arbitration Lost
ARBLST is set by hardware to inform application that CEC device is switching to reception due to arbitration lost event following the TXSOM command. ARBLST can be due either to a contending CEC device starting earlier or starting at the same time but with higher HEADER priority. After ARBLST assertion TXSOM bit keeps pending for next transmission attempt.
ARBLST is cleared by software write at 1.

Bit 6  **RXACKE**: Rx-Missing Acknowledge
In receive mode, RXACKE is set by hardware to inform application that no acknowledge was seen on the CEC line. RXACKE applies only for broadcast messages and in listen mode also for not directly addressed messages (destination address not enabled in OAR). RXACKE aborts message reception.
RXACKE is cleared by software write at 1.

Bit 5  **LBPE**: Rx-Long Bit Period Error
LBPE is set by hardware in case a Data-Bit waveform is detected with Long Bit Period Error. LBPE is set at the end of the maximum bit-extension tolerance allowed by RXTOL, in case falling edge is still longing. LBPE always stops reception of the CEC message. LBPE generates an Error-Bit on the CEC line if LBPEGEN=1. In case of broadcast, Error-Bit is generated even in case of LBPEGEN=0. LBPE is cleared by software write at 1.

Bit 4  **SBPE**: Rx-Short Bit Period Error
SBPE is set by hardware in case a Data-Bit waveform is detected with Short Bit Period Error. SBPE is set at the time the anticipated falling edge occurs. SBPE generates an Error-Bit on the CEC line. SBPE is cleared by software write at 1.
### 36.7.6 CEC interrupt enable register (CEC_IER)

Address offset: 0x14  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>TXACKIE</td>
<td>Tx-Missing Acknowledge Error Interrupt Enable</td>
</tr>
<tr>
<td>29</td>
<td>TXERRIE</td>
<td>Tx-Error Interrupt Enable</td>
</tr>
<tr>
<td>28</td>
<td>TXUDRIE</td>
<td>Tx-Underrun Interrupt Enable</td>
</tr>
<tr>
<td>27</td>
<td>TXACKIE</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>RXERRIE</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>RXUDRIE</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>RXACKIE</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>RXBRIE</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>RXOVRIE</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>RXENDIE</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>RXBRIE</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:13 Reserved, must be kept at reset value.

- **Bit 12 TXACKIE**: Tx-Missing Acknowledge Error Interrupt Enable  
  The TXACKIE bit is set and cleared by software.  
  0: TXACKE interrupt disabled  
  1: TXACKE interrupt enabled

- **Bit 11 TXERRIE**: Tx-Error Interrupt Enable  
  The TXERRIE bit is set and cleared by software.  
  0: TXERR interrupt disabled  
  1: TXERR interrupt enabled

- **Bit 10 TXUDRIE**: Tx-Underrun Interrupt Enable  
  The TXUDRIE bit is set and cleared by software.  
  0: TXUDR interrupt disabled  
  1: TXUDR interrupt enabled
Bit 9 TXENDIE: Tx-End Of Message Interrupt Enable
   The TXENDIE bit is set and cleared by software.
   0: TXEND interrupt disabled
   1: TXEND interrupt enabled

Bit 8 TXBRIE: Tx-Byte Request Interrupt Enable
   The TXBRIE bit is set and cleared by software.
   0: TXBR interrupt disabled
   1: TXBR interrupt enabled

Bit 7 ARBLSTIE: Arbitration Lost Interrupt Enable
   The ARBLSTIE bit is set and cleared by software.
   0: ARBLST interrupt disabled
   1: ARBLST interrupt enabled

Bit 6 RXACKIE: Rx-Missing Acknowledge Error Interrupt Enable
   The RXACKIE bit is set and cleared by software.
   0: RXACKE interrupt disabled
   1: RXACKE interrupt enabled

Bit 5 LBPEIE: Long Bit Period Error Interrupt Enable
   The LBPEIE bit is set and cleared by software.
   0: LBPE interrupt disabled
   1: LBPE interrupt enabled

Bit 4 SBPEIE: Short Bit Period Error Interrupt Enable
   The SBPEIE bit is set and cleared by software.
   0: SBPE interrupt disabled
   1: SBPE interrupt enabled

Bit 3 BREIE: Bit Rising Error Interrupt Enable
   The BREIE bit is set and cleared by software.
   0: BRE interrupt disabled
   1: BRE interrupt enabled

Bit 2 RXOVRIE: Rx-Buffer Overrun Interrupt Enable
   The RXOVRIE bit is set and cleared by software.
   0: RXOVR interrupt disabled
   1: RXOVR interrupt enabled

Bit 1 RXENDIE: End Of Reception Interrupt Enable
   The RXENDIE bit is set and cleared by software.
   0: RXEND interrupt disabled
   1: RXEND interrupt enabled

Bit 0 RXBRIE: Rx-Byte Received Interrupt Enable
   The RXBRIE bit is set and cleared by software.
   0: RXBR interrupt disabled
   1: RXBR interrupt enabled

Caution: (*) It is mandatory to write CEC_IER only when CECEN=0
36.7.7 HDMI-CEC register map

The following table summarizes the HDMI-CEC registers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CEC_CR</td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>CEC_CFG</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x08</td>
<td>CEC_TXDR</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0C</td>
<td>CEC_RXDR</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x10</td>
<td>CEC_ISR</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x14</td>
<td>CEC_IER</td>
<td>Reset value</td>
</tr>
</tbody>
</table>

Refer to Section 2.2 on page 54 for the register boundary addresses.
37 Debug support (DBG)

37.1 Overview

The STM32G0x1 devices are built around a Cortex®-M0+ core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state and the system's external state may be examined. Once examination is complete, the core and the system may be restored and program execution resumed.

The debug features are used by the debugger host when connecting to and debugging the STM32G0x1 MCUs.

One interface for debug is available:
- Serial wire

Figure 414. Block diagram of STM32G0x1 MCU and Cortex®-M0+-level debug support

1. The debug features embedded in the Cortex®-M0+ core are a subset of the Arm CoreSight Design Kit.

The Arm Cortex®-M0+ core provides integrated on-chip debug support. It is comprised of:
- SW-DP: Serial wire
- BPU: Break point unit
- DWT: Data watchpoint trigger
It also includes debug features dedicated to the STM32G0x1:

- Flexible debug pinout assignment
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

**Note:** For further information on debug functionality supported by the Arm Cortex®-M0+ core, refer to the Cortex®-M0+ Technical Reference Manual (see Section 37.2: Reference Arm documentation).

### 37.2 Reference Arm documentation

- Arm Debug Interface V5
- Arm CoreSight Design Kit revision r1p1 Technical Reference Manual

### 37.3 Pinout and debug port pins

The STM32G0x1 MCUs are available in various packages with different numbers of available pins.

#### 37.3.1 SWD port pins

Two pins are used as outputs for the SW-DP as alternate functions of general purpose I/Os. These pins are available on all packages.

<table>
<thead>
<tr>
<th>SW-DP pin name</th>
<th>SW debug port Type</th>
<th>Debug assignment</th>
<th>Pin assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWDIO</td>
<td>I/O</td>
<td>Serial Wire Data Input/Output</td>
<td>PA13</td>
</tr>
<tr>
<td>SWCLK</td>
<td>I</td>
<td>Serial Wire Clock</td>
<td>PA14</td>
</tr>
</tbody>
</table>

#### 37.3.2 SW-DP pin assignment

After reset (SYSRESETn or PORESETn), the pins used for the SW-DP are assigned as dedicated pins which are immediately usable by the debugger host.

However, the MCU offers the possibility to disable the SWD port and can then release the associated pins for general-purpose I/O (GPIO) usage. For more details on how to disable SW-DP port pins, please refer to Section 6.3.2: I/O pin alternate function multiplexer and mapping on page 198.
37.3.3 Internal pull-up & pull-down on SWD pins

Once the SW I/O is released by the user software, the GPIO controller takes control of these pins. The reset states of the GPIO control registers put the I/Os in the equivalent states:
- SWDIO: input pull-up
- SWCLK: input pull-down

_Having embedded pull-up and pull-down resistors removes the need to add external resistors._

37.4 ID codes and locking mechanism

There are several ID codes inside the MCU. ST strongly recommends the tool manufacturers (for example Keil, IAR, Raisonance) to lock their debugger using the MCU device ID located at address 0x40015800.

Only the DEV_ID[15:0] should be used for identification by the debugger/programmer tools (the revision ID must not be taken into account).

37.5 SWD port

37.5.1 SWD protocol introduction

This synchronous serial protocol uses two pins:
- SWCLK: clock from host to target
- SWDIO: bidirectional

The protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written to.

Bits are transferred LSB-first on the wire.

For SWDIO bidirectional management, the line must be pulled-up on the board (100 kΩ recommended by Arm).

Each time the direction of SWDIO changes in the protocol, a turnaround time is inserted where the line is not driven by the host nor the target. By default, this turnaround time is one bit time, however this can be adjusted by configuring the SWCLK frequency.

37.5.2 SWD protocol sequence

Each sequence consist of three phases:
1. Packet request (8 bits) transmitted by the host
2. Acknowledge response (3 bits) transmitted by the target
3. Data transfer phase (33 bits) transmitted by the host or the target
Refer to the Cortex®-M0+ TRM for a detailed description of DPACC and APACC registers.

The packet request is always followed by the turnaround time (default 1 bit) where neither the host nor target drive the line.

### Table 204. Packet request (8-bits)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Start</td>
<td>Must be “1”</td>
</tr>
<tr>
<td>1</td>
<td>APnDP</td>
<td>0: DP Access 1: AP Access</td>
</tr>
<tr>
<td>2</td>
<td>RnW</td>
<td>0: Write Request 1: Read Request</td>
</tr>
<tr>
<td>4:3</td>
<td>A[3:2]</td>
<td>Address field of the DP or AP registers (refer to Table 208 on page 1217)</td>
</tr>
<tr>
<td>5</td>
<td>Parity</td>
<td>Single bit parity of preceding bits</td>
</tr>
<tr>
<td>6</td>
<td>Stop</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Park</td>
<td>Not driven by the host. Must be read as “1” by the target because of the pull-up</td>
</tr>
</tbody>
</table>

The ACK Response must be followed by a turnaround time only if it is a READ transaction or if a WAIT or FAULT acknowledge has been received.

### Table 205. ACK response (3 bits)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0..2</td>
<td>ACK</td>
<td>001: FAULT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010: WAIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100: OK</td>
</tr>
</tbody>
</table>

The DATA transfer must be followed by a turnaround time only if it is a READ transaction.

### Table 206. DATA transfer (33 bits)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0..31</td>
<td>WDATA or RDATA</td>
<td>Write or Read data</td>
</tr>
<tr>
<td>32</td>
<td>Parity</td>
<td>Single parity of the 32 data bits</td>
</tr>
</tbody>
</table>

### 37.5.3 SW-DP state machine (reset, idle states, ID code)

The State Machine of the SW-DP has an internal ID code which identifies the SW-DP. It follows the JEP-106 standard. This ID code is the default Arm one and is set to 0x0BB11477 (corresponding to Cortex®-M0+).
Note: Note that the SW-DP state machine is inactive until the target reads this ID code.

- The SW-DP state machine is in RESET STATE either after power-on reset, or after the line is high for more than 50 cycles.
- The SW-DP state machine is in IDLE STATE if the line is low for at least two cycles after RESET state.
- After RESET state, it is mandatory to first enter into an IDLE state AND to perform a READ access of the DP-SW ID CODE register. Otherwise, the target will issue a FAULT acknowledge response on another transactions.

Further details of the SW-DP state machine can be found in the Cortex®-M0+ TRM and the CoreSight Design Kit r1p0 TRM.

37.5.4 DP and AP read/write accesses

- Read accesses to the DP are not posted: the target response can be immediate (if ACK=OK) or can be delayed (if ACK=WAIT).
- Read accesses to the AP are posted. This means that the result of the access is returned on the next transfer. If the next access to be done is NOT an AP access, then the DP-RDBUFF register must be read to obtain the result. The READOK flag of the DP-CTRL/STAT register is updated on every AP read access or RDBUFF read request to know if the AP read access was successful.
- The SW-DP implements a write buffer (for both DP or AP writes), that enables it to accept a write operation even when other transactions are still outstanding. If the write buffer is full, the target acknowledge response is “WAIT”. With the exception of IDCODE read or CTRL/STAT read or ABORT write which are accepted even if the write buffer is full.
- Because of the asynchronous clock domains SWCLK and HCLK, two extra SWCLK cycles are needed after a write transaction (after the parity bit) to make the write effective internally. These cycles should be applied while driving the line low (IDLE state).

This is particularly important when writing the CTRL/STAT for a power-up request. If the next transaction (requiring a power-up) occurs immediately, it will fail.

37.5.5 SW-DP registers

Access to these registers are initiated when APnDP=0

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Read</td>
<td>IDCODE</td>
<td></td>
<td>The manufacturer code is set to the default Arm code for Cortex®-M0+: 0x0BC11477 (identifies the SW-DP)</td>
</tr>
<tr>
<td>00</td>
<td>Write</td>
<td>ABORT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 207. SW-DP registers
### 37.5.6 SW-AP registers

Access to these registers are initiated when APnDP=1

There are many AP Registers addressed as the combination of:

- The shifted value A[3:2]
- The current value of the DP SELECT register.

#### Table 208. 32-bit debug port registers addressed through the shifted value A[3:2]

<table>
<thead>
<tr>
<th>Address</th>
<th>A[3:2] value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>00</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
| 0x4     | 01           | DP CTRL/STAT register. Used to:  
  - Request a system or debug power-up  
  - Configure the transfer operation for AP accesses  
  - Control the pushed compare and pushed verify operations.  
  - Read some status flags (overrun, power-up acknowledges) |
37.6 Core debug

Core debug is accessed through the core debug registers. Debug access to these registers is by means of the debug access port. It consists of four registers:

Table 209. Core debug registers

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHCSR</td>
<td>The 32-bit Debug Halting Control and Status Register</td>
</tr>
<tr>
<td></td>
<td>This provides status information about the state of the processor enable core debug halt and step the processor</td>
</tr>
<tr>
<td>DCRSR</td>
<td>The 17-bit Debug Core Register Selector Register:</td>
</tr>
<tr>
<td></td>
<td>This selects the processor register to transfer data to or from.</td>
</tr>
<tr>
<td>DCRDR</td>
<td>The 32-bit Debug Core Register Data Register:</td>
</tr>
<tr>
<td></td>
<td>This holds data for reading and writing registers to and from the processor selected by the DCRSR (Selector) register.</td>
</tr>
<tr>
<td>DEMCR</td>
<td>The 32-bit Debug Exception and Monitor Control Register:</td>
</tr>
<tr>
<td></td>
<td>This provides Vector Catching and Debug Monitor Control.</td>
</tr>
</tbody>
</table>

These registers are not reset by a system reset. They are only reset by a power-on reset. Refer to the Cortex®-M0+ TRM for further details.

To Halt on reset, it is necessary to:

- enable the bit0 (VC_CORRESET) of the Debug and Exception Monitor Control Register
- enable the bit0 (C_DEBUGEN) of the Debug Halting Control and Status Register

37.7 BPU (Break Point Unit)

The Cortex®-M0+ BPU implementation provides four breakpoint registers. The BPU is a subset of the Flash Patch and Breakpoint (FPB) block available in ARMv7-M (Cortex-M3 & Cortex-M4).
37.7.1 **BPU functionality**

The processor breakpoints implement PC based breakpoint functionality. Refer the ARmv6-M Arm and the Arm CoreSight Components Technical Reference Manual for more information about the BPU CoreSight identification registers, and their addresses and access types.

37.8 **DWT (Data Watchpoint)**

The Cortex®-M0+ DWT implementation provides two watchpoint register sets.

37.8.1 **DWT functionality**

The processor watchpoints implement both data address and PC based watchpoint functionality, a PC sampling register, and support comparator address masking, as described in the *ARMv6-M Arm*.

37.8.2 **DWT Program Counter Sample Register**

A processor that implements the data watchpoint unit also implements the ARmv6-M optional *DWT Program Counter Sample Register* (DWT_PCSR). This register permits a debugger to periodically sample the PC without halting the processor. This provides coarse grained profiling. See the *ARMv6-M Arm* for more information.

The Cortex®-M0+ DWT_PCSR records both instructions that pass their condition codes and those that fail.

37.9 **MCU debug component (DBG)**

The MCU debug component helps the debugger provide support for:

- Low-power modes
- Clock control for timers, watchdog and I2C during a breakpoint

37.9.1 **Debug support for low-power modes**

To enter low-power mode, the instruction WFI or WFE must be executed.

The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU.

The core does not allow FCLK or HCLK to be turned off during a debug session. As these are required for the debugger connection, during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In Sleep mode: FCLK and HCLK are still active. Consequently, this mode does not impose any restrictions on the standard debug features.
- In Stop/Standby mode, the DBG_STOP bit must be previously set by the debugger.

This enables the internal RC oscillator clock to feed FCLK and HCLK in Stop mode.
### 37.9.2 Debug support for timers, watchdog and I²C

During a breakpoint, it is necessary to choose how the counter of timers and watchdog should behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

For the I²C, the user can choose to block the SMBUS timeout during a breakpoint.

### 37.10 DBG registers

#### 37.10.1 DBG device ID code register (DBG_IDCODE)

The STM32G071xx and STM32G081xx products integrate a device ID code identifying the device and its die revision.

This code is accessible by the software debug port (two pins) or by the user software.

**DBG_IDCODE**

Address offset: 0x00

Only 32-bit access supported. Read-only

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

Bits 31:16 **REV_ID[15:0]** Revision identifier

This field indicates the revision of the device. Refer to Table 210.

Bits 15:12 Reserved: read 0b0110.

Bits 11:0 **DEV_ID[11:0]**: Device identifier

This field indicates the device ID. Refer to Table 210.

#### Table 210. DEV_ID and REV_ID field values

<table>
<thead>
<tr>
<th>Device</th>
<th>DEV_ID</th>
<th>Revision code</th>
<th>Revision number</th>
<th>REV_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32G071xx and STM32G081xx</td>
<td>0x460</td>
<td>A</td>
<td>1.0</td>
<td>0x1000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Z</td>
<td>1.1</td>
<td>0x1000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B</td>
<td>2.0</td>
<td>0x2000</td>
</tr>
<tr>
<td>STM32G031xx and STM32G041xx</td>
<td>0x466</td>
<td>A</td>
<td>1.0</td>
<td>0x1000</td>
</tr>
</tbody>
</table>
37.10.2  **DBG configuration register (DBG_CR)**

This register configures the low-power modes of the MCU under debug.

It is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

If the debugger host does not support this feature, it is still possible for the user software to write to this register.

Address offset: 0x04

POR Reset: 0x0000 0000 (not reset by system reset)

Only 32-bit access supported

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

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

Bit 2 **DBG_STANDBY**: Debug Standby and Shutdown modes

0: (FCLK=Off, HCLK=Off) The whole digital part is unpowered.

From software point of view, exiting Standby and Shutdown modes is identical as fetching reset vector (except a few status bit indicated that the MCU is resuming from Standby)

1: (FCLK=On, HCLK=On) In this case, the digital part is not powered and FCLK and HCLK are provided by the internal RC oscillator which remains active. In addition, the MCU generates a system reset during Standby and Shutdown modes so that exiting Standby and Shutdown is identical as fetching from reset.

Bit 1 **DBG_STOP**: Debug Stop mode

0: (FCLK=Off, HCLK=Off) In STOP mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the clock configuration is identical to the one after RESET (CPU clocked by the 8 MHz internal RC oscillator (HSI)). Consequently, the software must reprogram the clock controller to enable the PLL, the Xtal, and so on.

1: (FCLK=On, HCLK=On) In this case, when entering STOP mode, FCLK and HCLK are provided by the internal RC oscillator which remains active in STOP mode. When exiting STOP mode, the software must reprogram the clock controller to enable the PLL, the Xtal, and so on (in the same way it would do in case of DBG_STOP = 0)

37.10.3  **DBG APB freeze register 1 (DBG_APB_FZ1)**

This register configures the clocking of timers, RTC, IWDG, WWDG, and I2C SMBUS peripherals of the MCU under debug:

The register is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address offset: 0x08

Power on reset (POR): 0x0000 0000 (not reset by system reset)

Only 32-bit access are supported.
### Debug support (DBG)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td><strong>DBG_LPTIM1_STOP</strong>: Clocking of LPTIMER1 counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of LPTIMER1 when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td><strong>DBG_LPTIM2_STOP</strong>: Clocking of LPTIMER2 counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of LPTIMER2 when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td><strong>DBG_I2C1_SMBUS_TIMEOUT</strong>: SMBUS timeout when core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>0: Same behavior as in normal mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: The SMBUS timeout is frozen</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td><strong>DBG_IWDG_STOP</strong>: Clocking of IWDG counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of IWDG when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td><strong>DBG_WWDG_STOP</strong>: Clocking of WWDG counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of WWDG when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td><strong>DBG_RTC_STOP</strong>: Clocking of RTC counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of RTC when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td><strong>DBG_LPTIM1_STOP</strong>: Clocking of LPTIMER1 counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of LPTIMER1 when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td><strong>DBG_LPTIM2_STOP</strong>: Clocking of LPTIMER2 counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of LPTIMER2 when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td><strong>DBG_I2C1_SMBUS_TIMEOUT</strong>: SMBUS timeout when core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>0: Same behavior as in normal mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: The SMBUS timeout is frozen</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td><strong>DBG_IWDG_STOP</strong>: Clocking of IWDG counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of IWDG when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td><strong>DBG_WWDG_STOP</strong>: Clocking of WWDG counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of WWDG when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td><strong>DBG_RTC_STOP</strong>: Clocking of RTC counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of RTC when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td><strong>DBG_LPTIM1_STOP</strong>: Clocking of LPTIMER1 counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of LPTIMER1 when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td><strong>DBG_LPTIM2_STOP</strong>: Clocking of LPTIMER2 counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of LPTIMER2 when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td><strong>DBG_I2C1_SMBUS_TIMEOUT</strong>: SMBUS timeout when core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>0: Same behavior as in normal mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: The SMBUS timeout is frozen</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td><strong>DBG_IWDG_STOP</strong>: Clocking of IWDG counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of IWDG when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td><strong>DBG_WWDG_STOP</strong>: Clocking of WWDG counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of WWDG when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><strong>DBG_RTC_STOP</strong>: Clocking of RTC counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of RTC when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><strong>DBG_LPTIM1_STOP</strong>: Clocking of LPTIMER1 counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of LPTIMER1 when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>DBG_LPTIM2_STOP</strong>: Clocking of LPTIMER2 counter when the core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit enables/disables the clock to the counter of LPTIMER2 when the core is halted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Disable</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td><strong>DBG_I2C1_SMBUS_TIMEOUT</strong>: SMBUS timeout when core is halted</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>0: Same behavior as in normal mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: The SMBUS timeout is frozen</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **DBG_LPTIM1_STOP**: Clocking of LPTIMER1 counter when the core is halted
- This bit enables/disables the clock to the counter of LPTIMER1 when the core is halted:
  0: Enable
  1: Disable

Bit 30 **DBG_LPTIM2_STOP**: Clocking of LPTIMER2 counter when the core is halted
- This bit enables/disables the clock to the counter of LPTIMER2 when the core is halted:
  0: Enable
  1: Disable

Bits 29:22 Reserved, must be kept at reset value.

Bit 21 **DBG_I2C1_SMBUS_TIMEOUT**: SMBUS timeout when core is halted
- 0: Same behavior as in normal mode
- 1: The SMBUS timeout is frozen

Bits 20:13 Reserved, must be kept at reset value.

Bit 20 **DBG_IWDG_STOP**: Clocking of IWDG counter when the core is halted
- This bit enables/disables the clock to the counter of IWDG when the core is halted:
  0: Enable
  1: Disable

Bit 19 **DBG_WWDG_STOP**: Clocking of WWDG counter when the core is halted
- This bit enables/disables the clock to the counter of WWDG when the core is halted:
  0: Enable
  1: Disable

Bit 18 **DBG_RTC_STOP**: Clocking of RTC counter when the core is halted
- This bit enables/disables the clock to the counter of RTC when the core is halted:
  0: Enable
  1: Disable

Bits 9:6 Reserved, must be kept at reset value.
Bit 5 **DBG_TIM7_STOP**: Clocking of TIM7 counter when the core is halted.
   This bit enables/disables the clock to the counter of TIM7 when the core is halted:
   0: Enable
   1: Disable

Bit 4 **DBG_TIM6_STOP**: Clocking of TIM6 counter when the core is halted
   This bit enables/disables the clock to the counter of TIM6 when the core is halted:
   0: Enable
   1: Disable

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **DBG_TIM3_STOP**: Clocking of TIM3 counter when the core is halted
   This bit enables/disables the clock to the counter of TIM3 when the core is halted:
   0: Enable
   1: Disable

Bit 0 **DBG_TIM2_STOP**: Clocking of TIM2 counter when the core is halted
   This bit enables/disables the clock to the counter of TIM2 when the core is halted:
   0: Enable
   1: Disable

### 37.10.4 DBG APB freeze register 2 (DBG_APB_FZ2)

This register configures the clocking of timer counters when the MCU is under debug.

It is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address offset: 0x0C

POR: 0x0000 0000 (not reset by system reset)

Only 32-bit access is supported.
Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DBG_TIM17_STOP**: Clocking of TIM17 counter when the core is halted
   This bit enables/disables the clock to the counter of TIM17 when the core is halted:
   0: Enable
   1: Disable

Bit 17 **DBG_TIM16_STOP**: Clocking of TIM16 counter when the core is halted
   This bit enables/disables the clock to the counter of TIM16 when the core is halted:
   0: Enable
   1: Disable

Bit 16 **DBG_TIM15_STOP**: Clocking of TIM15 counter when the core is halted
   This bit enables/disables the clock to the counter of TIM15 when the core is halted:
   0: Enable
   1: Disable
   Only available on STM32G071xx and STM32G081xx, reserved on STM32G031xx and STM32G041xx.

Bit 15 **DBG_TIM14_STOP**: Clocking of TIM14 counter when the core is halted
   This bit enables/disables the clock to the counter of TIM14 when the core is halted:
   0: Enable
   1: Disable

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **DBG_TIM1_STOP**: Clocking of TIM1 counter when the core is halted
   This bit enables/disables the clock to the counter of TIM1 when the core is halted:
   0: Enable
   1: Disable

Bits 10:0 Reserved, must be kept at reset value.

### 37.10.5 **DBG register map**

The following table summarizes the Debug registers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Bits 31:19</th>
<th>Bits 26:17</th>
<th>Bits 16:9</th>
<th>Bits 8:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>DBG_IDCODE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REV_ID</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Reset value**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>DBG_IDCODE</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>DBG_CR</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Reset value**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>DBG_IDCODE</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>DBG_CR</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Reset value**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
<th>0x04</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>DBG_IDCODE</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td>REV_ID</td>
<td>DEV_ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>DBG_CR</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td>DBG_RUNNING</td>
<td>DBG_STOP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Reset value**
### Table 211. DBG register map and reset values (continued)

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x08   | DBG_APB_FZ1 |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM11_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM10_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM9_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM8_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM7_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM6_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM5_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM4_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM3_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM2_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM1_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM0_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x0C   | DBG_APB_FZ2 |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM17_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM16_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM15_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM14_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM13_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM12_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM11_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM10_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM9_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBG_TIM8_STOP |   0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

1. The reset value is product dependent. For more information, refer to Section 37.10.1: DBG device ID code register (DBG_IDCODE).

Refer to Section 2.2 on page 54 for the register boundary addresses.
38  **Device electronic signature**

The device electronic signature is stored in the System memory area of the Flash memory module, and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match to the characteristics of the STM32G0x1 microcontroller.

38.1  **Unique device ID register (96 bits)**

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as part of the security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0x1FFF 7590

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

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

**UID[31:16]**

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

**UID[15:0]**

Bits 31:0  **UID[31:0]**: X and Y coordinates on the wafer expressed in BCD format
Address offset: 0x04
Read only = 0xXXXX XXXX where X is factory-programmed

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID[63:48]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  **UID[63:40]**: LOT_NUM[23:0]
Lot number (ASCII encoded)

Bits 7:0  **UID[39:32]**: WAF_NUM[7:0]
Wafer number (8-bit unsigned number)

Address offset: 0x08
Read only = 0xXXXX XXXX where X is factory-programmed

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID[95:80]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **UID[95:64]**: LOT_NUM[55:24]
Lot number (ASCII encoded)

### 38.2 Flash memory size data register

Base address: 0x1FFF 75E0

Address offset: 0x00
Read only = 0xXXXX where X is factory-programmed

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

Bits 15:0  **FLASH_SIZE[15:0]**: Flash memory size
This bitfield indicates the size of the device Flash memory expressed in Kbytes.
As an example, 0x040 corresponds to 64 Kbytes.
38.3 Package data register

Base address: 0x1FFF 7500
Address offset: 0x00
Read only = 0xXXXX where X is factory-programmed

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

Bits 15:4 Reserved

Bits 3:0 **PKG[3:0]**: Package type

For STM32G071xx and STM32G081xx:
0000: UFQFPN28 general purpose (GP)
0001: UFQFPN28 Power Delivery (PD)
0100: UFQFPN32 / LQFP32 general purpose (GP)
0101: UFQFPN32 / LQFP32 Power Delivery (PD)
1000: UFQFPN48 / LQFP48
1100: LQPF64
Others: Reserved

For STM32G031xx and STM32G041xx:
0001: SO8N
0010: WLCSP18
0011: TSSOP20
0100: UFQFPN28
0101: UFQFPN32 / LQFP32
0111: UFQFPN48 / LQFP48
Others: Reserved
Revision history

Table 212. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>29-Oct-2018</td>
<td>1</td>
<td>Initial release.</td>
</tr>
<tr>
<td>02-May-2019</td>
<td>2</td>
<td>Integration of STM32G031xx and STM32G041xx, affecting:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 1.4: Availability of peripherals</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Figure 2: Memory map</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Table 3: STM32G031xx and STM32G041xx memory boundary addresses (added)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 2.3: Embedded SRAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 2.5: Boot configuration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 3.3: FLASH functional description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 3.4: FLASH option bytes (caution added in BOOT_LOCK bit definition)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section Table 6.: Flash memory organization (title modified)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 4: Power control (PWR) (indication of bits not available on STM32G031xx and STM32G041xx)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Figure 10: Clock tree</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 5: Reset and clock control (RCC) (indication of bits not available on STM32G031xx and STM32G041xx) and Section 5.2.12: Timer clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 7: System configuration controller (SYSCFG) (indication of bits not available on STM32G031xx and STM32G041xx)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 7.1.2: SYSCFG configuration register 2 (SYSCFG_CFRG2) (clamping diode enable bits added)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 9.1: Introduction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Table 36: DMA implementation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Table 41: DMAMUX instantiation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 11.3: Interrupt and exception vectors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 12: Extended interrupt and event controller (EXTI) (indication of bits not available on STM32G031xx and STM32G041xx)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Figure 163: General-purpose timer block diagram and Figure 186: External trigger input block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 26: Infrared interface (IRTIM)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Table 167: USART features</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Table 210: DEV_ID and REV_ID field values</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Table 37.10.4: DBG APB freeze register 2 (DBG_APB_FZ2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 38.3: Package data register</td>
</tr>
</tbody>
</table>
Index

A
ADC_AWD1TR .......................... 352
ADC_AWD2CR .......................... 356
ADC_AWD2TR .......................... 353
ADC_AWD3CR .......................... 358
ADC_AWD3TR .......................... 356
ADC_CALFACT ......................... 359
ADC_CCR .............................. 359
ADC_CFGR1 ........................... 346
ADC_CFGR2 ........................... 350
ADC_CHSELR .......................... 354-355
ADC_CR ............................... 344
ADC_DR ............................... 357
ADC_IER .............................. 342
ADC_ISR .............................. 340
ADC_SMPR ............................. 351
AES_CR ............................... 471
AES_DINR ............................. 475
AES_DOUTr ............................ 476
AES_IVR0 ............................. 478
AES_IVR1 ............................. 478
AES_IVR2 ............................. 478
AES_IVR3 ............................. 479
AES_KEYR0 ............................ 476
AES_KEYR1 ............................ 477
AES_KEYR2 ............................ 477
AES_KEYR3 ............................ 477
AES_KEYR4 ............................ 479
AES_KEYR5 ............................ 479
AES_KEYR6 ............................ 480
AES_KEYR7 ............................ 480
AES_SR ............................... 473
AES_SUSPxR ........................... 480

C
CEC_CFGR ................................ 1204
CEC_CR ................................ 1203
CEC_IER ................................ 1209
CEC_ISR ................................ 1207
CEC_RXDR .............................. 1207
CEC_TXDR .............................. 1207
COMP1_CSR ............................ 410
COMP2_CSR ............................ 412
CRC_CR ............................... 300
CRC_DR ............................... 299
CRC_IDR ............................... 299
CRC_INIT ............................. 300
CRC_POL ............................... 301

D
DAC_CCR ............................... 394
DAC_CR ................................ 383
DAC_DHR12L1 .......................... 387
DAC_DHR12L2 .......................... 389
DAC_DHR12LD .......................... 390
DAC_DHR12R1 .......................... 387
DAC_DHR12R2 .......................... 388
DAC_DHR12RD .......................... 390
DAC_DHR8R1 ........................... 388
DAC_DHR8R2 ........................... 389
DAC_DHR8RD ........................... 391
DAC_DOR1 ............................. 391
DAC_DOR2 ............................. 392
DAC_MCR ................................ 394
DAC_SHHR ............................. 397
DAC_SHRR ............................. 397
DAC_SHSR1 ............................ 396
DAC_SHSR2 ............................ 396
DAC_SR ............................... 392
DAC_SWTRGR ........................... 386
DBG_APB_FZ1 .......................... 1221
DBG_APB_FZ2 .......................... 1223
DBG_CR ............................... 1221
DBG_IDCODE .......................... 1220
DMA_CCRx ............................. 255
DMA_CMARx ............................. 259
DMA_CNDTRx ........................... 258
DMA_CPARx ............................. 258
DMA_IFCR ............................. 254
DMA_ISR .............................. 251
DMAMUX_CCFR .......................... 273
DMAMUX_CFR .......................... 273
DMAMUX_CSR .......................... 273
DMAMUX_CxCR .......................... 272
DMAMUX_RGCFR .......................... 275
DMAMUX_RGSR .......................... 275
DMAMUX_RGxCR .......................... 274
DMAMUX1_CFR .......................... 273

E
EXTI_EMR1 ............................ 292
EXTI_EMR2 ............................ 294
EXTI_EXTICRx .......................... 290
EXTI_FPR1 ............................ 290
<table>
<thead>
<tr>
<th>RM0444</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C_CR1</td>
<td>934</td>
</tr>
<tr>
<td>I2C_CR2</td>
<td>937</td>
</tr>
<tr>
<td>I2C_I2C</td>
<td>946</td>
</tr>
<tr>
<td>I2C_ISR</td>
<td>944</td>
</tr>
<tr>
<td>I2C_OAR1</td>
<td>940</td>
</tr>
<tr>
<td>I2C_OAR2</td>
<td>941</td>
</tr>
<tr>
<td>I2C_PECR</td>
<td>947</td>
</tr>
<tr>
<td>I2C_RXDR</td>
<td>948</td>
</tr>
<tr>
<td>I2C_TIMEOUT</td>
<td>943</td>
</tr>
<tr>
<td>I2C_TIMING</td>
<td>942</td>
</tr>
<tr>
<td>I2C_TXDR</td>
<td>948</td>
</tr>
<tr>
<td>I2Cx CR2</td>
<td>937</td>
</tr>
<tr>
<td>IWDG Kr</td>
<td>817</td>
</tr>
<tr>
<td>IWDG.xr</td>
<td>817</td>
</tr>
<tr>
<td>GPIOx_AF</td>
<td></td>
</tr>
</tbody>
</table>
### S

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPIx_CR1</td>
<td>1139</td>
</tr>
<tr>
<td>SPIx_CR2</td>
<td>1141</td>
</tr>
<tr>
<td>SPIx_CRCPR</td>
<td>1145</td>
</tr>
<tr>
<td>SPIx_DR</td>
<td>1145</td>
</tr>
<tr>
<td>SPIx_I2SCFGR</td>
<td>1146</td>
</tr>
<tr>
<td>SPIx_I2SPR</td>
<td>1148</td>
</tr>
<tr>
<td>SPIx_RXCRCR</td>
<td>1145</td>
</tr>
<tr>
<td>SYSCFG_ITLINE0</td>
<td>218</td>
</tr>
<tr>
<td>SYSCFG_ITLINE1</td>
<td>218</td>
</tr>
<tr>
<td>SYSCFG_ITLINE2</td>
<td>224</td>
</tr>
<tr>
<td>SYSCFG_ITLINE3</td>
<td>219</td>
</tr>
<tr>
<td>SYSCFG_ITLINE4</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE5</td>
<td>229</td>
</tr>
<tr>
<td>SYSCFG_ITLINE6</td>
<td>229</td>
</tr>
<tr>
<td>SYSCFG_ITLINE7</td>
<td>229</td>
</tr>
<tr>
<td>SYSCFG_ITLINE8</td>
<td>222</td>
</tr>
<tr>
<td>SYSCFG_ITLINE9</td>
<td>222</td>
</tr>
<tr>
<td>SYSCFG_ITLINE10</td>
<td>223</td>
</tr>
<tr>
<td>SYSCFG_ITLINE11</td>
<td>223</td>
</tr>
<tr>
<td>SYSCFG_ITLINE12</td>
<td>224</td>
</tr>
<tr>
<td>SYSCFG_ITLINE13</td>
<td>224</td>
</tr>
<tr>
<td>SYSCFG_ITLINE14</td>
<td>225</td>
</tr>
<tr>
<td>SYSCFG_ITLINE15</td>
<td>225</td>
</tr>
<tr>
<td>SYSCFG_ITLINE16</td>
<td>225</td>
</tr>
<tr>
<td>SYSCFG_ITLINE17</td>
<td>226</td>
</tr>
<tr>
<td>SYSCFG_ITLINE18</td>
<td>226</td>
</tr>
<tr>
<td>SYSCFG_ITLINE19</td>
<td>226</td>
</tr>
<tr>
<td>SYSCFG_ITLINE2</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE20</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE21</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE22</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE23</td>
<td>228</td>
</tr>
<tr>
<td>SYSCFG_ITLINE24</td>
<td>228</td>
</tr>
<tr>
<td>SYSCFG_ITLINE25</td>
<td>228</td>
</tr>
<tr>
<td>SYSCFG_ITLINE26</td>
<td>229</td>
</tr>
<tr>
<td>SYSCFG_ITLINE27</td>
<td>229</td>
</tr>
<tr>
<td>SYSCFG_ITLINE28</td>
<td>229</td>
</tr>
<tr>
<td>SYSCFG_ITLINE29</td>
<td>230</td>
</tr>
<tr>
<td>SYSCFG_ITLINE30</td>
<td>230-231</td>
</tr>
<tr>
<td>SYSCFG_ITLINE4</td>
<td>219</td>
</tr>
<tr>
<td>SYSCFG_ITLINE5</td>
<td>221</td>
</tr>
<tr>
<td>SYSCFG_ITLINE6</td>
<td>221</td>
</tr>
<tr>
<td>SYSCFG_ITLINE7</td>
<td>221</td>
</tr>
<tr>
<td>SYSCFG_ITLINE8</td>
<td>222</td>
</tr>
<tr>
<td>SYSCFG_ITLINE9</td>
<td>222</td>
</tr>
<tr>
<td>SYSCFG_ITLINE10</td>
<td>223</td>
</tr>
</tbody>
</table>

### T

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP_BKPxR</td>
<td>881</td>
</tr>
<tr>
<td>TAMP_CCR1</td>
<td>874</td>
</tr>
<tr>
<td>TAMP_CCR2</td>
<td>875</td>
</tr>
<tr>
<td>TAMP_FLTCR</td>
<td>876</td>
</tr>
<tr>
<td>TAMP_IER</td>
<td>877</td>
</tr>
<tr>
<td>TAMP_MISR</td>
<td>879</td>
</tr>
<tr>
<td>TAMP_SCR</td>
<td>880</td>
</tr>
<tr>
<td>TAMP_SR</td>
<td>878</td>
</tr>
<tr>
<td>SYSCFG_CFRG1</td>
<td>213</td>
</tr>
<tr>
<td>SYSCFG_CFRG2</td>
<td>216</td>
</tr>
<tr>
<td>SYSCFG_ITLINE0</td>
<td>218</td>
</tr>
<tr>
<td>SYSCFG_ITLINE1</td>
<td>218</td>
</tr>
<tr>
<td>SYSCFG_ITLINE2</td>
<td>224</td>
</tr>
<tr>
<td>SYSCFG_ITLINE3</td>
<td>219</td>
</tr>
<tr>
<td>SYSCFG_ITLINE4</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE5</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE6</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE7</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE8</td>
<td>222</td>
</tr>
<tr>
<td>SYSCFG_ITLINE9</td>
<td>222</td>
</tr>
<tr>
<td>SYSCFG_ITLINE10</td>
<td>223</td>
</tr>
</tbody>
</table>

---

**Index RM0444**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPIx_SR</td>
<td>1143</td>
</tr>
<tr>
<td>SPIx_TXCRCR</td>
<td>1146</td>
</tr>
<tr>
<td>SPIx_RXCRCR</td>
<td>1145</td>
</tr>
<tr>
<td>SYSCFG_CFRG1</td>
<td>213</td>
</tr>
<tr>
<td>SYSCFG_CFRG2</td>
<td>216</td>
</tr>
<tr>
<td>SYSCFG_ITLINE0</td>
<td>218</td>
</tr>
<tr>
<td>SYSCFG_ITLINE1</td>
<td>218</td>
</tr>
<tr>
<td>SYSCFG_ITLINE2</td>
<td>224</td>
</tr>
<tr>
<td>SYSCFG_ITLINE3</td>
<td>224</td>
</tr>
<tr>
<td>SYSCFG_ITLINE4</td>
<td>225</td>
</tr>
<tr>
<td>SYSCFG_ITLINE5</td>
<td>225</td>
</tr>
<tr>
<td>SYSCFG_ITLINE6</td>
<td>225</td>
</tr>
<tr>
<td>SYSCFG_ITLINE7</td>
<td>226</td>
</tr>
<tr>
<td>SYSCFG_ITLINE8</td>
<td>226</td>
</tr>
<tr>
<td>SYSCFG_ITLINE9</td>
<td>226</td>
</tr>
<tr>
<td>SYSCFG_ITLINE10</td>
<td>223</td>
</tr>
<tr>
<td>SYSCFG_ITLINE11</td>
<td>223</td>
</tr>
<tr>
<td>SYSCFG_ITLINE12</td>
<td>224</td>
</tr>
<tr>
<td>SYSCFG_ITLINE13</td>
<td>224</td>
</tr>
<tr>
<td>SYSCFG_ITLINE14</td>
<td>225</td>
</tr>
<tr>
<td>SYSCFG_ITLINE15</td>
<td>225</td>
</tr>
<tr>
<td>SYSCFG_ITLINE16</td>
<td>225</td>
</tr>
<tr>
<td>SYSCFG_ITLINE17</td>
<td>226</td>
</tr>
<tr>
<td>SYSCFG_ITLINE18</td>
<td>226</td>
</tr>
<tr>
<td>SYSCFG_ITLINE19</td>
<td>226</td>
</tr>
<tr>
<td>SYSCFG_ITLINE2</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE20</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE21</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE22</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE23</td>
<td>228</td>
</tr>
<tr>
<td>SYSCFG_ITLINE24</td>
<td>228</td>
</tr>
<tr>
<td>SYSCFG_ITLINE25</td>
<td>228</td>
</tr>
<tr>
<td>SYSCFG_ITLINE26</td>
<td>229</td>
</tr>
<tr>
<td>SYSCFG_ITLINE27</td>
<td>229</td>
</tr>
<tr>
<td>SYSCFG_ITLINE28</td>
<td>229</td>
</tr>
<tr>
<td>SYSCFG_ITLINE29</td>
<td>230</td>
</tr>
<tr>
<td>SYSCFG_ITLINE30</td>
<td>230-231</td>
</tr>
<tr>
<td>SYSCFG_ITLINE4</td>
<td>219</td>
</tr>
<tr>
<td>SYSCFG_ITLINE5</td>
<td>221</td>
</tr>
<tr>
<td>SYSCFG_ITLINE6</td>
<td>221</td>
</tr>
<tr>
<td>SYSCFG_ITLINE7</td>
<td>221</td>
</tr>
<tr>
<td>SYSCFG_ITLINE8</td>
<td>222</td>
</tr>
<tr>
<td>SYSCFG_ITLINE9</td>
<td>222</td>
</tr>
<tr>
<td>SYSCFG_ITLINE10</td>
<td>223</td>
</tr>
<tr>
<td>SYSCFG_ITLINE11</td>
<td>223</td>
</tr>
<tr>
<td>SYSCFG_ITLINE12</td>
<td>224</td>
</tr>
<tr>
<td>SYSCFG_ITLINE13</td>
<td>224</td>
</tr>
<tr>
<td>SYSCFG_ITLINE14</td>
<td>225</td>
</tr>
<tr>
<td>SYSCFG_ITLINE15</td>
<td>225</td>
</tr>
<tr>
<td>SYSCFG_ITLINE16</td>
<td>225</td>
</tr>
<tr>
<td>SYSCFG_ITLINE17</td>
<td>226</td>
</tr>
<tr>
<td>SYSCFG_ITLINE18</td>
<td>226</td>
</tr>
<tr>
<td>SYSCFG_ITLINE19</td>
<td>226</td>
</tr>
<tr>
<td>SYSCFG_ITLINE2</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE20</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE21</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE22</td>
<td>227</td>
</tr>
<tr>
<td>SYSCFG_ITLINE23</td>
<td>228</td>
</tr>
<tr>
<td>SYSCFG_ITLINE24</td>
<td>228</td>
</tr>
<tr>
<td>SYSCFG_ITLINE25</td>
<td>228</td>
</tr>
<tr>
<td>SYSCFG_ITLINE26</td>
<td>229</td>
</tr>
<tr>
<td>SYSCFG_ITLINE27</td>
<td>229</td>
</tr>
<tr>
<td>SYSCFG_ITLINE28</td>
<td>229</td>
</tr>
<tr>
<td>SYSCFG_ITLINE29</td>
<td>230</td>
</tr>
<tr>
<td>SYSCFG_ITLINE30</td>
<td>230-231</td>
</tr>
</tbody>
</table>

---

1232/1234 RM0444 Rev 2
TIM1_CCMR2 .......................... 561-562
TIM1_CCMR3 ................................ 576
TIM1_CCR1 ................................ 568
TIM1_CCR2 ................................ 569
TIM1_CCR3 ................................ 569
TIM1_CCR5 ................................ 577
TIM1_CCR6 ................................ 578
TIM1_CNT .................................. 567
TIM1_CR1 .................................. 546
TIM1_CR2 .................................. 547
TIM1_DCR .................................. 574
TIM1_DIER .................................. 552
TIM1_DMAR .................................. 575
TIM1_EGR .................................. 556
TIM1_OR1 .................................. 576
TIM1_PSC .................................. 567
TIM1_RCR .................................. 568
TIM1_SMCR .................................. 550
TIM1_SR .................................. 554
TIM1_TISEL .................................. 582
TIM14_ARR .................................. 695
TIM14_CCMR1 ............................ 691-692
TIM14_CCR1 ................................ 696
TIM14_CNT .................................. 695
TIM14_CR1 .................................. 688
TIM14_DIER .................................. 689
TIM14_EGR .................................. 690
TIM14_PSC .................................. 695
TIM14_SR .................................. 689
TIM14_TISEL .................................. 696
TIM15_AF1 .................................. 760
TIM15_ARR .................................. 754
TIM15_BDTR .................................. 756
TIM15_CCER .................................. 751
TIM15_CCMR1 ............................ 747-748
TIM15_CCR1 .................................. 755
TIM15_CCR2 .................................. 756
TIM15_CNT .................................. 754
TIM15_CR1 .................................. 739
TIM15_CR2 .................................. 740
TIM15_DCR .................................. 759
TIM15_DIER .................................. 743
TIM15_DMAR .................................. 759
TIM15_EGR .................................. 746
TIM15_PSC .................................. 754
TIM15_RCR .................................. 755
TIM15_SMCR .................................. 742
TIM15_SR .................................. 744
TIM15_TISEL .................................. 761
TIM16_AF1 ................................. 781-782
TIM16_TISEL .................................. 784
TIM17_AF1 .................................. 784
TIM17_TISEL .................................. 785
TIM2_AF1 .................................. 653
TIM2_OR1 .................................. 653
TIM2_TISEL .................................. 654
TIM3_AF1 .................................. 654
TIM3_OR1 .................................. 653
TIM3_TISEL .................................. 655
TIMx_ARR ................................. 649, 671, 775
TIMx_BDTR .................................. 777
TIMx_CCER ................................. 640, 642, 769-770
TIMx_CCMR1 ............................. 640-645
TIMx_CCMR2 ............................. 640-645
TIMx_CCR1 .................................. 649, 776
TIMx_CCR2 .................................. 650
TIMx_CCR3 .................................. 650
TIMx_CCR4 ................................. 649-776
TIMx_CNT ................................. 647-648, 670, 774
TIMx_CR1 ................................. 630, 667, 764
TIMx_CR2 ................................. 631, 669, 765
TIMx_DCR .................................. 652, 780
TIMx_DIER .................................. 636, 669, 766
TIMx_DMAR .................................. 652, 780
TIMx_EGR .................................. 639, 670, 768
TIMx_PSC ................................. 648, 671, 775
TIMx_RCCR .................................. 677
TIMx_RCR .................................. 677
TIMx_SMCR .................................. 633
TIMx_SR .................................. 637, 670, 767
USART_BRR ................................ 1019
USART_CR1 ................................ 1003, 1007
USART_CR2 ................................ 1010
USART_CR3 ................................ 1014
USART_GTPR ................................ 1019
USART_ICR ................................ 1033
USART_ISR ................................ 1022, 1028
USART_PRES ................................ 1036
USART_RDR ................................ 1035
USART_RQR ................................ 1021
USART_RTOR ................................ 1020
USART_TDR ................................ 1035
VREFBUF_CCR ................................ 403
VREFBUF_CSR ................................ 402
WWDG_CFR ................................ 827
WWDG_CR ................................ 826
WWDG_SR ................................ 828